

Wright State University

CORE Scholar

---

[Browse all Theses and Dissertations](#)

[Theses and Dissertations](#)

---

2017

## Extended Kalman Filter and LQR controller design for quadrotor UAVs

Muneeb Masood Raja  
*Wright State University*

Follow this and additional works at: [https://corescholar.libraries.wright.edu/etd\\_all](https://corescholar.libraries.wright.edu/etd_all)



Part of the [Electrical and Computer Engineering Commons](#)

---

### Repository Citation

Raja, Muneeb Masood, "Extended Kalman Filter and LQR controller design for quadrotor UAVs" (2017).  
*Browse all Theses and Dissertations*. 1761.  
[https://corescholar.libraries.wright.edu/etd\\_all/1761](https://corescholar.libraries.wright.edu/etd_all/1761)

This Thesis is brought to you for free and open access by the Theses and Dissertations at CORE Scholar. It has been accepted for inclusion in Browse all Theses and Dissertations by an authorized administrator of CORE Scholar. For more information, please contact [library-corescholar@wright.edu](mailto:library-corescholar@wright.edu).

EXTENDED KALMAN FILTER AND LQR CONTROLLER  
DESIGN FOR QUADROTOR UAVs

A thesis submitted in partial fulfillment of the  
requirements for the degree of  
Master of Science in Electrical Engineering

By

Muneeb Masood Raja  
B.M.E., National University of Sciences and Technology, Pakistan, 2014

---

2017  
Wright State University

WRIGHT STATE UNIVERSITY

GRADUATE SCHOOL

May 30, 2017

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY  
SUPERVISION BY Muneeb Masood Raja ENTITLED Extended Kalman Filter and  
LQR controller design for quadrotor UAVs BE ACCEPTED IN PARTIAL  
FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of  
Science in Electrical Engineering.

---

Xiaodong Zhang, Ph.D.  
Thesis Advisor

---

Brian Rigling, Ph.D.  
Chair, Department of Electrical Engineering

Committee on  
Final Examination

---

Xiaodong Zhang, Ph.D.

---

Kuldip Rattan, Ph.D.

---

Jonathan Muse, Ph.D.

---

Robert E.W. Fyffe, Ph.D.  
Vice President for Research and  
Dean of the Graduate School

## ABSTRACT

Raja, Muneeb Masood. M.S.E.E., Department of Electrical Engineering, Wright State University, 2017. Extended Kalman Filter and LQR controller design for Quadrotor UAVs.

A quadrotor is a unique class of UAVs with vertical take off and landing (VTOL) capability and has attracted significant attention due to its importance in various applications. This thesis presents the design and experimental implementation of Extended Kalman Filters (EKFs) to estimate the states of a quadrotor and a Linear Quadratic Regulator (LQR) controller with integral action to meet the desired control objectives. In case of the Extended Kalman Filters, two different situations are considered: (1) all the states including the Inertial Measurement Unit (IMU) biases are estimated; (2) only the attitude, altitude, and vertical velocity are estimated. The second case is added as a safety feature to provide enough feedback signals to stabilize and land the quadrotor in the event of a position measurement loss, e.g. from a GPS due to jamming. A double loop control structure is implemented using an LQR controller with integral action, the inner loop contains the attitude and the altitude control, and the outer loop consists of x and y translational positions control. Finally, some preliminary results on the integration of C codes with Simulink using C MEX S-functions is described. A C library of a laser rangefinder sensor is transferred to a C MEX S-function to generate a 2D map of the environment using the laser sensor distance measurements to identify obstacles present within the range of the sensor. The concept of multi-threading and the integration of pthread library with Simulink using C MEX S-function are also described.



## TABLE OF CONTENTS

1.	INTRODUCTION .....	1
	1.1. Literature Survey .....	1
	1.2. Research Objectives .....	2
	1.3. Thesis Organization .....	2
2.	Quadrotor Model and Experimental Platform .....	4
	2.1. Quadrotor Dynamic Model .....	4
	2.1.1 Reference Frames .....	4
	2.1.2 Euler Angles and Quaternions .....	5
	2.1.3 Nonlinear Quadrotor Model .....	6
	2.1.4 Motor Model .....	8
	2.2. Experimental Setup .....	9
3.	Extended Kalman Filter .....	11
	3.1. Introduction .....	11
	3.2. General Overview on Kalman Filters .....	11
	3.3. Problem Formulation .....	13
	3.3.1 Full State Estimator With IMU biases .....	13
	3.3.2 Prediction Model .....	14
	3.3.3 Measurement Model .....	15
	3.3.4 Estimating Altitude, Vertical Velocity and Euler Angles .....	16
	3.4. Experimental Results .....	17
	3.4.1 Full State Estimator: Case Without Added Biases .....	17
	3.4.2 Full State Estimator: Case With Added Biases .....	20
	3.4.3 Estimating Euler Angles, Altitude and Vertical Velocity .....	24
	3.5. Conclusion .....	26
4.	Linear Quadratic Regulator .....	28
	4.1. Introduction .....	28

4.2. General Overview of the Linear Quadratic Regulator .....	28
4.3. Control System Design .....	30
4.3.1 Control Architecture .....	30
4.3.2 Control Law Design Specifications .....	32
4.3.3 Linearization .....	32
4.3.4 Attitude Control .....	34
4.3.5 Altitude control .....	35
4.3.6 X and Y Position Control .....	36
4.4. Simulation Results.....	38
4.5. Experimental Results .....	43
4.6. Conclusion .....	47
5. OBSTACLE DETECTION AND C MEX S-FUNCTIONS.....	48
5.1. Introduction.....	48
5.2. Problem Formulation.....	48
5.2.1 Obstacle Detection .....	48
5.2.2 C MEX S-functions .....	49
5.3. Preliminary Results .....	51
5.4. Conclusion .....	52
6. CONCLUSION AND FUTURE RESEARCH .....	53
6.1. Summary of Results .....	53
6.2. Future Research .....	54
BIBLIOGRAPHY .....	56

## LIST OF FIGURES

2.1	Quadrotor reference frames .....	4
2.2	Experimental setup .....	10
3.1	Inertial positions estimates .....	18
3.2	Euler angles estimates .....	19
3.3	Body velocity estimates using Euler based model .....	19
3.4	Inertial velocity estimates using quaternion based model .....	20
3.5	Inertial positions estimates with added biases .....	21
3.6	Euler angles estimates with added biases .....	22
3.7	Body frame velocity estimates using Euler model with added biases .....	22
3.8	Inertial velocity estimates with added biases using quaternion model .....	23
3.9	Accelerometer biases estimate .....	23
3.10	Gyroscope biases estimate .....	24
3.11	Euler angles estimates .....	25
3.12	Altitude and vertical velocity estimate .....	26
4.1	Control loop architecture .....	30
4.2	LQR with integral block diagram .....	30
4.3	LQR double loop control architecture .....	31
4.4	LQR inner loop block architecture .....	31
4.5	Simulink Model for the LQR controller .....	38
4.6	Step Response of Euler angles (Simulation Result) .....	39
4.7	Step response of altitude (Simulation Result) .....	39
4.8	Step Response of x-y positions (Simulation Result) .....	40
4.9	Step Response of Euler angles (Simulation Result) .....	40
4.10	Step response of altitude (Simulation Result) .....	41
4.11	x-y position control using LQR (Simulation Result) .....	42
4.12	Attitude control using LQR (Simulation Result) .....	42

4.13	Altitude control using LQR (Simulation Result).....	43
4.14	Euler angles control using LQR .....	44
4.15	x-y position control using LQR.....	44
4.16	Altitude control using LQR.....	45
4.17	Angular Velocities .....	45
4.18	Linear Velocities .....	46
4.19	Actuator Signals .....	46
5.1	Laser rangefinder scan .....	49
5.2	Articheture of a C MEX S-function .....	49
5.3	Proposed Experimental Model .....	50
5.4	2D point cloud of the surroundings .....	51
5.5	2D map of the surroundings .....	52

## LIST OF TABLES

4.1	Eigen values, damping ratio, and natural frequency from attitude controller.....	35
4.2	Eigen values, damping ratio, and natural frequency from altitude controller .....	36
4.3	Eigen values, damping ratio, and natural frequency from x-y position controller ..	37

## ACKNOWLEDGMENTS

I am greatly indebted to and would like to express my profound gratitude to my advisor, Dr. Xiaodong Zhang, for being a source of constant support and guidance for me, not only during the project but throughout my Masters. His door was always open to me whenever I needed help in any form. It is only through his belief in me and his guidance that I was able to complete all the desired work.

I would also like to thank Dr. Jonathan Muse for his valuable insight and recommendations. His guidance and support helped me reach the results at a faster pace and helped me understand the project deeply.

Further, I would take the opportunity to thank the worthy committee member Dr. Kuldeep Rattan for his feedback and attendance at the defense.

I am blessed to have a beautiful supportive family, which is always standing behind my back and being a source of motivation. I would specifically like to acknowledge my brother, Hamza Masood Rajas efforts to push me to go for a Masters and excel wherever I go.

In the end, I would like to thank my beautiful mother for being the pillar of strength and hope. It is only because of her ability to stand against all odds that I stand here today. Thank you ammi, nothing would have been possible without you.

## 1. INTRODUCTION

### 1.1 LITERATURE SURVEY

Unmanned Aerial Vehicles (UAVs) as the name suggests are airborne vehicles which operates without any pilot under remote or autonomous circumstances. Due to its increasing use in military and civil applications for surveillance, reconnaissance, search and rescue missions, it has become a research topic for numerous research groups such as GTMax of Georgia Institute of Technology UAV [1], ServoHeli family of Shenyang Institute of Automation in Chinese Academy of Sciences [2], and Lincoln Beaver Works UAV Systems Center in Massachusetts Institute of Technology [3]. A major advantage that a UAV has over manned aerial vehicles is that its flight time is restricted only by fuel/battery life, whereas in manned aerial vehicles the human factors like fatigue have to be considered [4].

UAVs provide increased autonomy, which can enable humans to delegate those tasks that can be done more effectively by a computer, including synchronizing activities between multiple unmanned systems, software agents and war-fighters, thus freeing humans to focus on more complex decision making [5]. Since autonomous operation doesn't require any human intervention, autonomous control of UAVs are much more challenging. Due to this reason, UAVs are more susceptible to mishaps and crashes as compared to manned aerial vehicles. In order to avoid these failures and to enhance autonomy, advanced and robust control techniques need to be devised and implemented.

A quadrotor is a unique class of UAVs with characteristics like vertical take off and landing (VTOL) capability, the ability to hover, and their compact structures allow them to navigate to places where humans can't intervene. The fast dynamics which makes the quadrotor maneuverable also require quick and accurate state estimates to perform necessary control actions. In order to serve this purpose, the Kalman Filter (KF) is widely used by researchers for accurate state estimates especially for applications where the sensor measurements are noisy or doesn't provide state information directly. In order to apply the traditional linear KF on nonlinear systems, the Extended Kalman Filter (EKF) was developed. Researchers have been applying this algorithm in several different ways to obtain a variety of state estimates. For instance, [6] implements an EKF based on a drag force enhanced model to estimate all the states of a quadrotor along with the unknown drag coefficient, [7, 8] estimates only the attitude of the quadrotor using

the Inertial Measurement Unit (IMU) sensor, and [9] tracks the target by implementing a vision based EKF. To satisfy the control objectives, various control methods for a quadrotor UAVs have been proposed, including PID control [10], robust  $H_\infty$  control [11], back-stepping control [12], Linear Quadratic Regulator (LQR) control [13], and so on. Compared with a PID controller, an LQR controller is difficult to design and needs access to all the states. In the literature, the LQR controller is mostly restricted only to simulation studies or stabilization of limited states [14, 15].

## 1.2 RESEARCH OBJECTIVES

This research aims at the design and practical implementation of the Extended Kalman Filter due to its importance to advanced guidance, navigation, and control objectives. Furthermore, in order to achieve the control requirements, an LQR controller with integral action is designed and implemented on the experimental setup. The EKF is designed for two different circumstances: (1) the case when the position measurements are available; (2) the case when the position measurements are not available. In the first case, IMU biases are also considered in the EKF estimates to cater for any sensor errors. The second case can be considered as a safety feature in the event we lose position measurements. Additionally, an LQR controller is designed for both the inner and outer loop control by following a double loop control architecture. For the case when position measurements are available, both outer and inner loop controls are established, whereas only inner loop control is designed for the case when position measurements are lost. The nonlinear quadrotor system model is linearized before designing the gains for the LQR controller, but the simulations result are produced using the nonlinear quadrotor dynamics. In order to validate the performance of the algorithms, flight tests are conducted using a real-time test environment to prove the effectiveness of the designed techniques.

## 1.3 THESIS ORGANIZATION

The remaining of the thesis is organized as follows:

- **Chapter 2** describes the reference frames involved in the quadrotor dynamics, rotation matrices in different representations, nonlinear mathematical model, and



the real-time experimental platform used to implement the designed algorithms.

- **Chapter 3** provides a detailed explanation of state estimation using Extended Kalman Filter (EKF) in two scenarios: (1) position measurements are available and IMU biases are also considered and estimated; (2) position measurements are not available and IMU biases are not taken into account. In the first case, all quadrotor states including the IMU biases are estimated using both the Euler and the quaternions approaches, whereas in the latter case, only the Euler angles, altitude and vertical velocities are estimated.
- **Chapter 4** presents the design of the Linear Quadratic Regulator (LQR) controller with integral action to satisfy the control objectives of the quadrotor. An LQR controller is designed based on the linearized quadrotor system dynamics, which is also discussed in this chapter. Simulation as well as real-time flight results are shown. The feedback signals in the case of real-time flight are provided by the EKF designed in Chapter 3, whereas the feedback for the case of simulation studies are generated by integrating the nonlinear system dynamic equations.
- **Chapter 5** describes the importance of C MEX S-functions used to integrate C programs and libraries in Simulink. A Hokuyo laser rangefinder sensor library provided by the manufacturers and written in C programming language is transferred to the C MEX S-function, and a 2D sensor scan data is saved and plotted appropriately to visualize the obstacles present in the surroundings. Additionally, a routine using pthread library is run in the C MEX S-function, which can later be used for the efficient implementation of computationally expensive tasks required by obstacle detection and avoidance algorithms.
- **Chapter 6** includes some concluding remarks and some discussions of possible future research directions.

## 2. QUADROTOR MODEL AND EXPERIMENTAL PLATFORM

In this chapter, various reference coordinate systems used by the quadrotor, transformation between different frames using rotation matrices, dynamic model of the quadrotor, and the real-time experimental system setup is covered.

### 2.1 QUADROTOR DYNAMIC MODEL

#### 2.1.1 REFERENCE FRAMES

There are several coordinate frames involved while developing quadrotor's dynamic system model [16,17]. Figure 2.1 shows three of the reference frames whose description is given below.

1. Inertial frame ( $F_i$ ) with  $x_i$  directed towards North,  $y_i$  towards East, and  $z_i$  Down with respect to the earth. It is also referred to as the earth reference frame.
2. Body frame ( $F_b$ ) is fixed on the body of the quadrotor with the origin being the center of gravity.  $x_b$  points towards the nose of the quadrotor,  $y_b$  points towards the right wing, and  $z_b$  points to the ground.
3. Vehicle frame axis ( $F_v$ ) are aligned with the axis of inertial frame, whereas, the center is at the center of the mass of quadrotor.

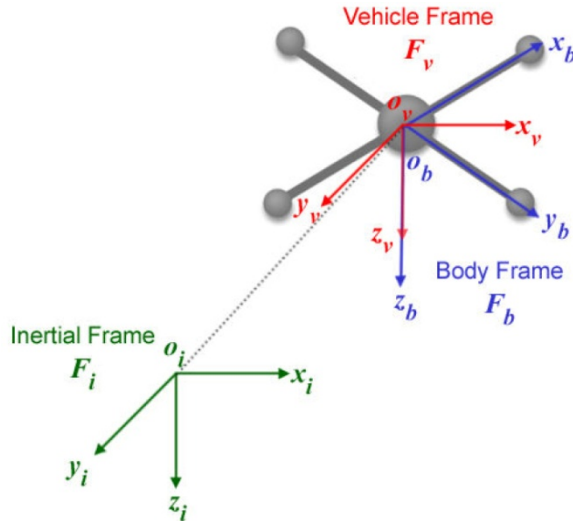


Figure 2.1: Reference frames of quadrotor

There are two intermediate reference frames in between vehicle and body frames. The first intermediate frame ( $F_1$ ) is obtained by rotating the vehicle frame about  $z_v$  to

generate a yaw angle  $\psi$ . Similarly, rotating the first intermediate frame ( $F_1$ ) about  $y_{F_1}$  by a pitch angle  $\theta$  takes us to the second intermediate frame ( $F_2$ ). Finally, rotating the quadrotor about  $x_{F_2}$  by a roll angle  $\phi$  gives us the body frame.

### 2.1.2 EULER ANGLES AND QUATERNIONS

Euler angles are used to represent the relative orientation of a coordinate frame with respect to another coordinate frame as a series of three rotations typically represented by  $\phi$ ,  $\theta$  and  $\psi$  [18]. There are a total of twelve possible rotation sequences that can be used to describe the orientation of two coordinate frames.

Transformation from the vehicle frame to  $F_1$  requires a rotation  $\psi$  about the  $z_v$  axis and is given by:

$$R_v^1(\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (2.1)$$

Similarly, transformation of  $F_1$  to the  $F_2$  is done by rotating  $y_{F_1}$  with an angle  $\theta$  about y-axis. This transformation is given by the following rotation matrix:

$$R_1^2(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix}. \quad (2.2)$$

Finally, the rotation  $\phi$  represents the rotation about  $x_{F_2}$  axis and transforms the second intermediate frame to the body frame. The rotation matrix representing this transformation is given below:

$$R_2^b(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix}. \quad (2.3)$$

So, the vehicle and body frame coordinate frames are related by the following rotation matrix:

$$R_b^e(\phi, \theta, \psi) = R_v^1(\psi) R_1^2(\theta) R_2^b(\phi)$$

$$R_b^e(\psi, \theta, \phi) = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix}, \quad (2.4)$$

where  $c\phi \triangleq \cos(\phi)$ ,  $s\phi \triangleq \sin(\phi)$ , etc.

Since, the inertial and vehicle frames are aligned to each other, so (2.4) represents both the rotation from body frame to the inertial and vehicle reference frames.

In case of Euler angles, a phenomenon known as gimbal lock effect can occur in which a degree of freedom is lost when the pitch angle becomes  $90^\circ$  in case of the 3-2-1 rotation sequence. In order to avoid this singularity, quaternions are used [19, 20]. Just like Euler angles, quaternions can also be used to provide the spacial orientation of any rigid body. Euler angles can be converted to its equivalent representation in quaternions using the following equation:

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} \cos(\frac{\phi}{2})\cos(\frac{\theta}{2})\cos(\frac{\psi}{2}) + \sin(\frac{\phi}{2})\sin(\frac{\theta}{2})\sin(\frac{\psi}{2}) \\ \sin(\frac{\phi}{2})\cos(\frac{\theta}{2})\cos(\frac{\psi}{2}) - \cos(\frac{\phi}{2})\sin(\frac{\theta}{2})\sin(\frac{\psi}{2}) \\ \cos(\frac{\phi}{2})\sin(\frac{\theta}{2})\cos(\frac{\psi}{2}) + \sin(\frac{\phi}{2})\cos(\frac{\theta}{2})\sin(\frac{\psi}{2}) \\ \cos(\frac{\phi}{2})\cos(\frac{\theta}{2})\sin(\frac{\psi}{2}) - \sin(\frac{\phi}{2})\sin(\frac{\theta}{2})\cos(\frac{\psi}{2}) \end{bmatrix}. \quad (2.5)$$

Similarly, rotation matrix represented by (2.4) can be written in terms of quaternions as shown below:

$$R_b^e(\psi, \theta, \phi) = \begin{bmatrix} q_1^2 + q_2^2 - q_3^2 - q_4^2 & 2(q_2q_3 - q_1q_4) & 2(q_1q_3 + q_2q_4) \\ 2(q_2q_3 + q_1q_4) & q_1^2 - q_2^2 + q_3^2 - q_4^2 & 2(q_3q_4 - q_1q_2) \\ 2(q_2q_4 - q_1q_3) & 2(q_1q_2 + q_3q_4) & q_1^2 - q_2^2 - q_3^2 + q_4^2 \end{bmatrix}. \quad (2.6)$$

### 2.1.3 NONLINEAR QUADROTOR MODEL

A quadrotor has a total of twelve state variables defined below:

$$states = [p_{x_i} \ p_{y_i} \ p_{z_i} \ u \ v \ w \ \phi \ \theta \ \psi \ p \ q \ r]^T, \quad (2.7)$$

where  $p_{x_i}$ ,  $p_{y_i}$  and  $p_{z_i}$  represents quadrotor position in the inertial frame;  $u$ ,  $v$  and  $w$  are the quadrotor's velocity expressed in the body frame;  $\phi$ ,  $\theta$  and  $\psi$  are the roll, pitch and yaw angles of the quadrotor; and  $p$ ,  $q$  and  $r$  are the angular rates of the quadrotor.

The quadrotor system model is derived using the Newton-Euler equations of motion. Solving for these equations of motion results in the state space model represented by the following equations.

$$\begin{bmatrix} \dot{p}_{x_i} \\ \dot{p}_{y_i} \\ \dot{p}_{z_i} \end{bmatrix} = R_b^e(\psi, \theta, \phi) \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad (2.8)$$

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} + \begin{bmatrix} -g\sin(\theta) \\ g\cos(\theta)\sin(\phi) \\ g\cos(\theta)\cos(\phi) \end{bmatrix} + \frac{1}{m} \begin{bmatrix} 0 \\ 0 \\ -F \end{bmatrix} \quad (2.9)$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} (\frac{I_y - I_z}{I_x})qr \\ (\frac{I_z - I_x}{I_y})pr \\ (\frac{I_x - I_y}{I_z})pq \end{bmatrix} + \begin{bmatrix} (\frac{1}{I_x})\tau_\phi \\ (\frac{1}{I_y})\tau_\theta \\ (\frac{1}{I_z})\tau_\psi \end{bmatrix} \quad (2.10)$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = R_\rho(\phi, \theta) \begin{bmatrix} p \\ q \\ r \end{bmatrix}, \quad (2.11)$$

where  $R_b^e$  is the rotation matrix shown by (2.4),  $F$  is the thrust force,  $m$  is the mass of quadrotor,  $g$  is the acceleration due to gravity,  $I_x$ ,  $I_y$  and  $I_z$  are the moment of inertia terms about the x, y, and z axis of the body frame,  $\tau_\phi$ ,  $\tau_\theta$  and  $\tau_\psi$  are the rolling torque, pitching torque, and the yawing torque provided as inputs to the system,  $R_\rho(\phi, \theta)$  shows the relationship between angular rates and the Euler angles rates which is shown below.

$$R_\rho(\phi, \theta) = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)\sec(\theta) & \cos(\phi)\sec(\theta) \end{bmatrix}. \quad (2.12)$$

#### 2.1.4 MOTOR MODEL

Each rotor of the quadrotor produces a force acting opposite to the z-axis of the body frame. Similarly, these rotating rotors also produce torques causing roll, pitch and yaw actions. These forces and torques are directly proportional to the velocity square of the rotors and can be represented by the following expressions.

$$F_i = k_f \Omega_i^2 \quad (2.13)$$

$$T_i = -k_t \text{sgn}(\Omega_i) \Omega_i^2, \quad (2.14)$$

where the  $i$  subscript represents the quantity associated with the  $i^{th}$  rotor, with  $i = 1, 2, 3, 4$ ,  $\Omega_i$  is the  $i^{th}$  rotor velocity,  $k_f$  and  $k_t$  are the force and torque constants;  $\text{sgn}$  captures the fact that the rotors spin in different directions depending on the location.

The forces acting on the quadrotor can be represented in terms of the rotational velocities of the rotors with the help of a mapping matrix  $M$  as follows:

$$\begin{bmatrix} F \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = M \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix}, \quad (2.15)$$

where the mapping matrix  $M$  for the "X" configuration is defined as [21]:

$$M = \begin{bmatrix} k_f & k_f & k_f & k_f \\ \frac{k_f d}{\sqrt{2}} & -\frac{k_f d}{\sqrt{2}} & -\frac{k_f d}{\sqrt{2}} & \frac{k_f d}{\sqrt{2}} \\ \frac{k_f d}{\sqrt{2}} & \frac{k_f d}{\sqrt{2}} & -\frac{k_f d}{\sqrt{2}} & -\frac{k_f d}{\sqrt{2}} \\ k_t & -k_t & k_t & -k_t \end{bmatrix}, \quad (2.16)$$

where  $\Omega_i^2$  is the square of the  $i^{th}$  rotor velocity, and  $d$  is the distance from the center of the mass of the quadrotor to the center of the rotor.

## 2.2 EXPERIMENTAL SETUP

The general layout of the experimental setup used during this research comprises of the following three major components.

- Quadrotor built and assembled inside the lab using off the shelf components.
- Vicon motion capture system.
- Ground station computer system.

The Vicon camera system [22] provides the positions and Euler angles of the quadrotor during real-time flights at a sample rate of 100 Hz. A TCP/IP communication protocol is used to receive/send data between the ground station and the on-board microcontroller. The quadrotor has a Gumstix DuoVero Zephyr microcontroller with integrated wireless communication capability on-board, which can process data up to 1 GHz and have a 1 GB memory. The module also provide Inertial Measurement Unit (IMU), temperature, and battery voltage measurements. The whole control algorithm was developed in MATLAB and Simulink, which is built and converted to C/C++ code, and fed to the on-board computer for real-time testing of the designed algorithm.

The required Vicon signals coming from the motion capture system, and the IMU signals from the on-board sensor are used as measurements in the Extended Kalman Filter algorithm developed in this thesis. The results are then used in the LQR controller design to enhance the performance of the controller for real-time flights of the quadrotor. The parameters/gains of the LQR controllers can be changed during run time, if necessary, and the control actions can be switched in between outer loop and inner loop control to check the performance and robustness of both the Kalman Filters described in Chapter 3 and the LQR controller. The Simulink model is also equipped with a traditional PID controller designed as described in [21], and the operator of the ground station can also switch between the PID and LQR controller during run-time, in case one of them doesn't provide satisfactory control performance.

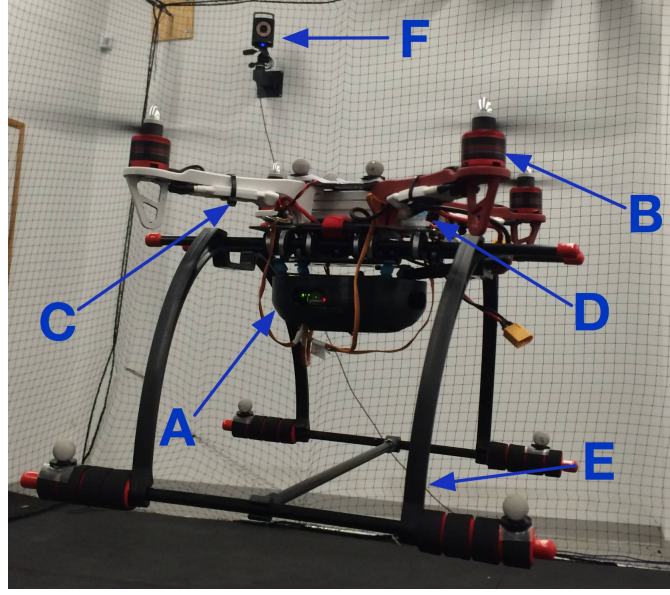


Figure 2.2: Experimental Plattform of the Quadrotor [21]

Figure 2.2 shows the experimental platform in real-time flight used in this thesis. Main components consists of the Qbrain embedded controller (A), BLDC motors attached to the fours propellers (B), four ESC's attached to the four motors (C), which changes the rotational velocities of the motors depending on the PWM signal coming from the controller, the landing gear attached to the quadrotor for safe landing (E), and a 3-cell 12 V DC voltage battery with 2000mAh capacity (D).



### 3. Extended Kalman Filter

#### 3.1 INTRODUCTION

In this section, a full-state Extended Kalman Filter, estimating all states of the quadrotor along with the accelerometer and gyroscope biases is discussed. In case of outdoor flights, GPS is widely used to provide position measurements, which is susceptible to issues like jamming. If anything like this happens during flight, it is highly likely that the quadrotor will crash. In order to avoid this, another Kalman Filter is implemented, estimating only the Euler angles, altitude and vertical velocity of the quadrotor. For testing purposes, this Kalman Filter uses yaw angle and altitude measurements from the Vicon camera system, but after installing magnetometer and sonar sensors, this Kalman Filter can be made independent of the Vicon camera system and will rely only on the on-board sensors. These two Kalman Filters are working in parallel to each other and can switch between each other depending on the requirements. During normal operation, i.e. when Vicon or GPS signals are available, the full state Kalman Filter is used to provide feedback signals, and outer loop control is established. Once the Vicon signals are lost, the feedback for the controller comes from the latter Kalman Filter. Now, in this case only the Euler angles and altitude of the quadrotor are controlled to ensure that the quadrotor flies and lands safely.

#### 3.2 GENERAL OVERVIEW ON KALMAN FILTERS

Kalman Filter which is also known as linear quadratic estimation is a very widely used tool over the past few decades for guidance, navigation and control of unmanned aerial vehicles and is very effective for estimating the states of an aircraft. The idea of Kalman filter was first given by R.E. Kalman in his paper about linear filtering [23]. Since then, it has been subject to vast research and application. A general introduction to Kalman Filters can be found in [24], whereas more extensive references and texts related to Kalman Filter can be found in [25–27]. An extended Kalman Filter (EKF) is one of the modifications of a Kalman Filter which uses the nonlinear system model to estimate the systems states. The EKF can be divided into two major parts: the prediction and correction step. In the prediction phase, the states are propagated forward using the nonlinear continuous time model of the system to give us the state prediction.

Whereas, sensor measurements are available in a discrete time interval during the correction phase, and the process moves forward in a discrete manner giving us the corrected state estimates. The general algorithm for EKF estimation is given below [28].

### 1. Prediction Phase (Time Update)

Prediction update uses the following expressions.

$$\hat{x}^- = f(\hat{x}, u) \quad (3.1)$$

$$P^- = AP + PA^T + Q, \quad (3.2)$$

where  $f(\hat{x}, u)$  is the nonlinear system model;  $P$  is the predicted error covariance matrix;  $Q$  is the positive definite noise process covariance matrix; and  $A$  is defined as a Jacobian matrix represented as

$$A = \frac{\partial f}{\partial x}(\hat{x}, u). \quad (3.3)$$

### 2. Correction Phase (Measurement Update)

When a measurement is available from a sensor, the EKF updates the states, error covariance matrix, and Kalman gain using the following equations:

$$K = P^- C^T (R + C P^- C^T)^{-1} \quad (3.4)$$

$$\hat{x} = \hat{x}^- + K(y - h(\hat{x}^-)) \quad (3.5)$$

$$P = (I - KC)P^-, \quad (3.6)$$

where  $K$  is the Kalman gain;  $R$  is the noise covariance associated with the sensors;  $\hat{x}$  and  $P$  are the current estimates of the state and error covariance which will be used in the prediction phase for the next sampling time. The terms with minus superscript represents the values coming from the prediction phase.  $C$  is the Jacobian of the measurement vector with respect to the state vector and can be represented as shown below:

$$C = \frac{\partial h}{\partial x}(\hat{x}^-, u). \quad (3.7)$$

The system model should be accurate enough to generate good state estimates, because an inaccurate process model could result in poor estimates. Nevertheless, an Extended Kalman Filter (EKF) when used carefully can result in reliable state estimates.

### 3.3 PROBLEM FORMULATION

#### 3.3.1 FULL STATE ESTIMATOR WITH IMU BIASES

For this problem, states to be estimated for the quadrotor, when attitude is represented in quaternions are shown below:

$$x = [p_{x_i} \ p_{y_i} \ p_{z_i} \ v_{x_i} \ v_{y_i} \ v_{z_i} \ q_1 \ q_2 \ q_3 \ q_4 \ \beta_{i_b} \ \beta_{j_b} \ \beta_{k_b} \ \alpha_{i_b} \ \alpha_{j_b} \ \alpha_{k_b}]^T \quad (3.8)$$

and when attitude is represented in Euler angles, states take the following form:

$$x = [p_{x_i} \ p_{y_i} \ p_{z_i} \ u \ v \ w \ \phi \ \theta \ \psi \ \beta_{i_b} \ \beta_{j_b} \ \beta_{k_b} \ \alpha_{i_b} \ \alpha_{j_b} \ \alpha_{k_b}]^T, \quad (3.9)$$

where  $p_{x_i}$ ,  $p_{y_i}$  and  $p_{z_i}$  represents quadrotor position in inertial frame,  $v_{x_i}$ ,  $v_{y_i}$  and  $v_{z_i}$  are the velocity of the quadrotor expressed in inertial frame,  $u$ ,  $v$  and  $w$  are the quadrotor's velocity represented in body frame,  $q_1$ ,  $q_2$ ,  $q_3$  and  $q_4$  is the attitude expressed in quaternions,  $\phi$ ,  $\theta$  and  $\psi$  are the roll, pitch and yaw angles of quadrotor,  $\beta_{i_b}$ ,  $\beta_{j_b}$  and  $\beta_{k_b}$  are the biases in gyroscope represented in body frame, and  $\alpha_{i_b}$ ,  $\alpha_{j_b}$  and  $\alpha_{k_b}$  are the biases associated with the accelerometer measurements.

The following sensor measurements are available to us:

$$Y = [\gamma_{i_b} \ \gamma_{j_b} \ \gamma_{k_b} \ \delta_{i_b} \ \delta_{j_b} \ \delta_{k_b} \ p_{x_v} \ p_{y_v} \ p_{z_v} \ \psi_v]^T, \quad (3.10)$$

where  $\gamma_{i_b}$ ,  $\gamma_{j_b}$  and  $\gamma_{k_b}$  represents the gyroscope measurements,  $\delta_{i_b}$ ,  $\delta_{j_b}$  and  $\delta_{k_b}$  are the accelerometer measurements,  $p_{x_v}$ ,  $p_{y_v}$  and  $p_{z_v}$  are the true inertial positions; and  $\psi_v$  is the true yaw angle of the quadrotor. The gyroscope and accelerometer measurements are provided by the on-board IMU sensor, whereas, vicon motion capture camera system provide the true inertial positions and yaw angle.

The accelerometer and the gyroscope measurements are used as direct inputs to

the system model. They are represented as follows:

$$\mathbf{u}_{imu} = \begin{bmatrix} \gamma_{i_b} \\ \gamma_{j_b} \\ \gamma_{k_b} \\ \delta_{i_b} \\ \delta_{j_b} \\ \delta_{k_b} \end{bmatrix} = \begin{bmatrix} p + \beta_{i_b} \\ q + \beta_{j_b} \\ r + \beta_{k_b} \\ a_x + \alpha_{i_b} \\ a_y + \alpha_{j_b} \\ a_z + \alpha_{k_b} \end{bmatrix}, \quad (3.11)$$

where  $p$ ,  $q$  and  $r$  are the true angular rates, and  $a_x$ ,  $a_y$  and  $a_z$  are the true quadrotor accelerations. All the other parameters are already defined under the description of (3.8) - (3.10).

### 3.3.2 PROCESS (PREDICTION) MODEL

Again, we will have two prediction models, i.e. attitude expressed in quaternions and Euler angles.

#### 1. Quaternions Based Prediction Model

Sixteen state equations constitute the quaternion based prediction model, given below:

$$f(\hat{x}, u) = \begin{bmatrix} \hat{v}_{x_i} \\ \hat{v}_{y_i} \\ \hat{v}_{z_i} \\ (\hat{q}_1^2 + \hat{q}_2^2 - \hat{q}_3^2 - \hat{q}_4^2)a_x + 2(\hat{q}_2\hat{q}_3 - \hat{q}_1\hat{q}_4)a_y + 2(\hat{q}_2\hat{q}_4 + \hat{q}_1\hat{q}_3)a_z \\ (\hat{q}_1^2 - \hat{q}_2^2 + \hat{q}_3^2 - \hat{q}_4^2)a_y + 2(\hat{q}_2\hat{q}_3 + \hat{q}_1\hat{q}_4)a_x + 2(\hat{q}_3\hat{q}_4 - \hat{q}_1\hat{q}_2)a_z \\ (\hat{q}_1^2 - \hat{q}_2^2 - \hat{q}_3^2 + \hat{q}_4^2)a_z + 2(\hat{q}_2\hat{q}_4 - \hat{q}_1\hat{q}_3)a_x + 2(\hat{q}_3\hat{q}_4 + \hat{q}_1\hat{q}_2)a_y + g \\ -0.5(p\hat{q}_2 + q\hat{q}_3 + r\hat{q}_4) \\ 0.5(p\hat{q}_1 - q\hat{q}_4 + r\hat{q}_3) \\ 0.5(p\hat{q}_4 + q\hat{q}_1 - r\hat{q}_2) \\ 0.5(-p\hat{q}_3 + q\hat{q}_2 + r\hat{q}_1) \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \quad (3.12)$$

## 2. Euler Angle Based Prediction Model

The following equations constitute the prediction model of a quadrotor for this case.

$$f(\hat{x}, u) = \begin{bmatrix} \hat{u}(c_{\hat{\theta}}c_{\hat{\psi}}) + \hat{v}(s_{\hat{\phi}}s_{\hat{\theta}}c_{\hat{\psi}} - c_{\hat{\phi}}s_{\hat{\psi}}) + \hat{w}(c_{\hat{\phi}}s_{\hat{\theta}}c_{\hat{\psi}} + s_{\hat{\phi}}s_{\hat{\psi}}) \\ \hat{u}(c_{\hat{\theta}}s_{\hat{\psi}}) + \hat{v}(s_{\hat{\phi}}s_{\hat{\theta}}s_{\hat{\psi}} + c_{\hat{\phi}}c_{\hat{\psi}}) + \hat{w}(c_{\hat{\phi}}s_{\hat{\theta}}s_{\hat{\psi}} - s_{\hat{\phi}}c_{\hat{\psi}}) \\ \hat{u}(-s_{\hat{\theta}}) + \hat{v}(s_{\hat{\phi}}c_{\hat{\theta}}) + \hat{w}(c_{\hat{\phi}}c_{\hat{\theta}}) \\ r\hat{v} - q\hat{w} - g\hat{s}\hat{\theta} + a_x \\ p\hat{w} - r\hat{u} + g\hat{c}\hat{\theta}s_{\hat{\phi}} + a_y \\ q\hat{u} - p\hat{v} + g\hat{c}\hat{\theta}c_{\hat{\phi}} + a_z \\ p + q(s_{\hat{\phi}}\tan\hat{\theta}) + r(c_{\hat{\phi}}\tan\hat{\theta}) \\ q(c_{\hat{\phi}}) - r(s_{\hat{\phi}}) \\ q(\frac{s_{\hat{\phi}}}{c_{\hat{\theta}}}) + r(\frac{c_{\hat{\phi}}}{c_{\hat{\theta}}}) \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad (3.13)$$

where  $c_{\hat{\phi}} \triangleq \cos\hat{\phi}$ ,  $s_{\hat{\phi}} \triangleq \sin\hat{\phi}$ ,  $c_{\hat{\theta}} \triangleq \cos\hat{\theta}$ ,  $s_{\hat{\theta}} \triangleq \sin\hat{\theta}$ ,  $c_{\hat{\psi}} \triangleq \cos\hat{\psi}$ , and  $s_{\hat{\psi}} \triangleq \sin\hat{\psi}$

The hat on the top of each state is used to identify them as estimated states, which are computed by the Extended Kalman Filter routine. This prediction function is then differentiated with respect to the states to compute A matrix as given by (3.3), which represent the linear estimate for how the states changes with time.

### 3.3.3 MEASUREMENT MODEL

The measurement update model, i.e.  $y$  as shown in (3.5) would end up to be the following.

$$y = [p_{x_v} \ p_{y_v} \ p_{z_v} \ \psi_v \ \delta_{i_b} \ \delta_{j_b}]^T. \quad (3.14)$$

The x and y axis accelerations can be dropped out, but having these two in the equation makes it easier to approximate the biases associated with it. The Jacobian matrix for the measurement update are found in a similar manner as that of the prediction

model. This matrix represented by  $C$  is computed using (3.7).

### 3.3.4 ESTIMATING ALTITUDE, VERTICAL VELOCITY AND EULER ANGLES

For this case, states to be estimated are given below:

$$x_1 = [p_{z_i} \ v_{z_i} \ \phi \ \theta \ \psi]^T, \quad (3.15)$$

where  $p_{z_i}$  represents the quadrotor's altitude,  $v_{z_i}$  is the vertical velocity of the quadrotor expressed in inertial frame, and  $\phi$ ,  $\theta$  and  $\psi$  are the roll, pitch and yaw angles of quadrotor.

During the implementation of this Kalman Filter, we assume that we only have the following sensor measurements available.

$$Y = [\gamma_{i_b} \ \gamma_{j_b} \ \gamma_{k_b} \ \delta_{i_b} \ \delta_{j_b} \ \delta_{k_b} \ p_{sonar} \ \psi_{mag}]^T, \quad (3.16)$$

where  $\gamma_{i_b}$ ,  $\gamma_{j_b}$  and  $\gamma_{k_b}$  represents the gyroscope measurements,  $\delta_{i_b}$ ,  $\delta_{j_b}$  and  $\delta_{k_b}$  are the accelerometer measurements,  $p_{sonar}$  is the measured altitude ; and  $\psi_{mag}$  is the measured yaw angle of the quadrotor. The altitude and yaw angle measurements are taken from the Vicon camera system, but after installing the sonar and magnetometer sensors for altitude and yaw measurements, Vicon provided measurements can be replaced. The gyroscope and accelerometer measurements are provided by the on-board IMU sensor. Although, the IMU readings will have inherent biases included in its measurements, they are assumed to be zero during the implementation of this Kalman Filter.

This Extended Kalman Filter is implemented in two phases.

1. Euler angles are estimated first.
2. These estimated Euler angles are then used to estimate the remaining two states, i.e. altitude and vertical velocity.

The following two prediction models are used during implementation of the two phases given above:

$$f_1(\hat{x}, u) = \begin{bmatrix} \gamma_{i_b} + \gamma_{j_b}(s_{\hat{\phi}} \tan \hat{\theta}) + \gamma_{k_b}(c_{\hat{\phi}} \tan \hat{\theta}) \\ \gamma_{j_b}(c_{\hat{\phi}}) - \gamma_{k_b}(s_{\hat{\phi}}) \\ \gamma_{j_b}(\frac{s_{\hat{\phi}}}{c_{\hat{\theta}}}) + \gamma_{k_b}(\frac{c_{\hat{\phi}}}{c_{\hat{\theta}}}) \end{bmatrix} \quad (3.17)$$

and

$$f_2(\hat{x}, u) = \begin{bmatrix} \hat{v}_{z_i} \\ -s_{\hat{\theta}}\delta_{i_b} + s_{\hat{\phi}}c_{\hat{\theta}}\delta_{j_b} + c_{\hat{\phi}}c_{\hat{\theta}}\delta_{k_b} + g \end{bmatrix}. \quad (3.18)$$

The hat on top of each state means that its the estimated value derived as an output of the Kalman Filter. The Euler angles estimated from the first stage Kalman Filter are used as an input in the second prediction model, as can be seen from (3.18).

The gyroscope sensor measurements are used as direct inputs in the attitude process model, whereas in the altitude and vertical velocity estimation, accelerometer measurements are used as direct inputs to the process model.

### 3.4 EXPERIMENTAL RESULTS

In this section, experimental results are shown for the state estimates generated by the Extended Kalman Filter implemented on a real time system (quadrotor). The positions and the reference Euler angles are derived from the Vicon motion capture camera system and used as the actual truth values. Similarly, an on-board IMU is used to provide us the necessary angular rates and the accelerations. In order to verify the bias estimates of accelerometer and gyroscope, a constant bias is added to them at different times during the flight. Additionally, the Extended Kalman Filter experimental results estimating only the Euler angles, altitude and vertical velocity are shown. This estimate is not dependent on the Vicon measurements and can independently estimate the required states by only relying on the on-board sensors. In the event that we lose the position measurements coming from the Vicon cameras, without the use of this Kalman Filter to estimate altitude and Euler angles required for the inner loop control, the quadrotor might crash. So, we develop this safety feature added to the quadrotor during real time flights.

#### 3.4.1 FULL STATE ESTIMATOR: CASE WITHOUT ADDED BIASES

In this case, the quadrotor is flying under normal circumstances without the presence of any added sensor faults/biases. Note that there are some constant biases already present in the gyroscope and accelerometer readings, which are also estimated and catered for accordingly. Figure 3.1 - 3.2 shows the estimation results of inertial

positions and Euler angles with solid blue lines showing the true values of the parameters, red lines representing the Kalman Filter estimate based on the model with attitude represented in terms of Euler angles, and the yellow lines representing the quaternion based Kalman Filter estimates. Additionally, Figure 3.3 shows the velocity estimate in the body frame generated by the Euler based implementation, whereas Figure 3.4 represents the inertial frame velocity estimate as a result of quaternion based Kalman Filter execution.

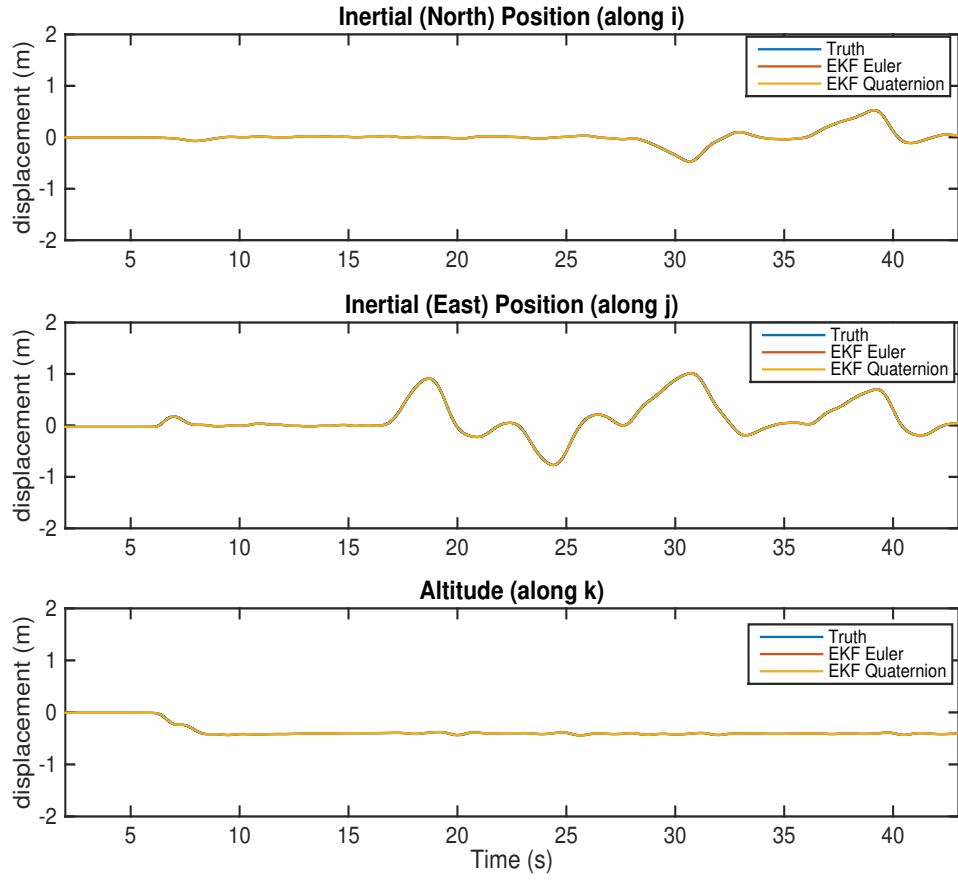


Figure 3.1: Estimates of inertial positions



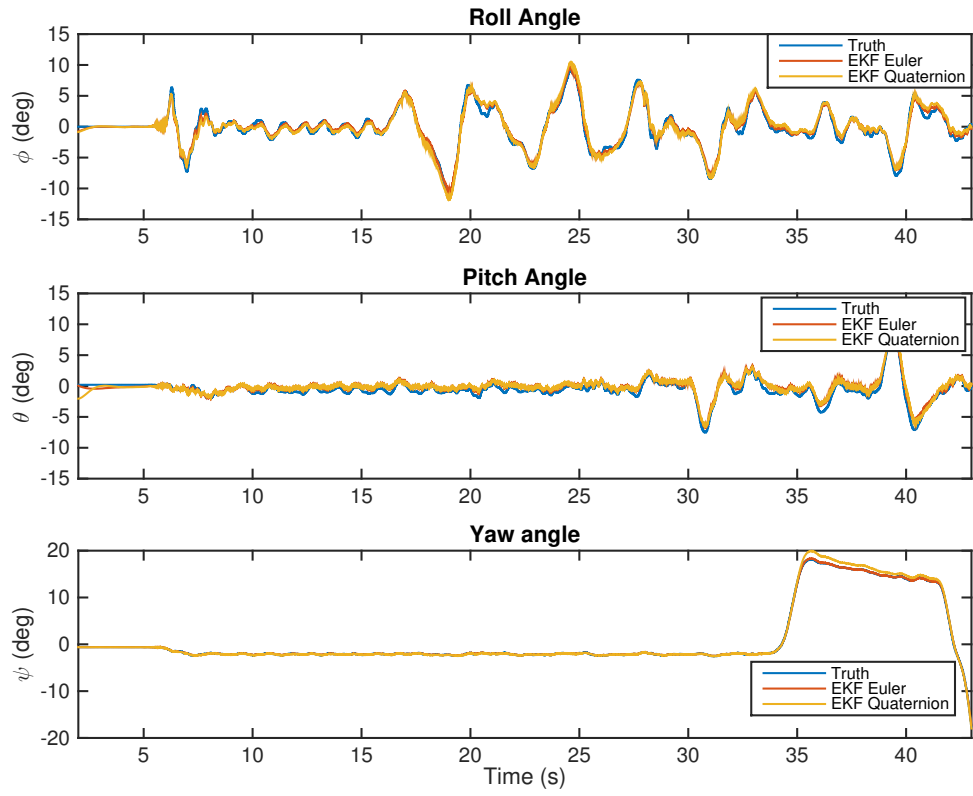


Figure 3.2: Estimates of Euler angles

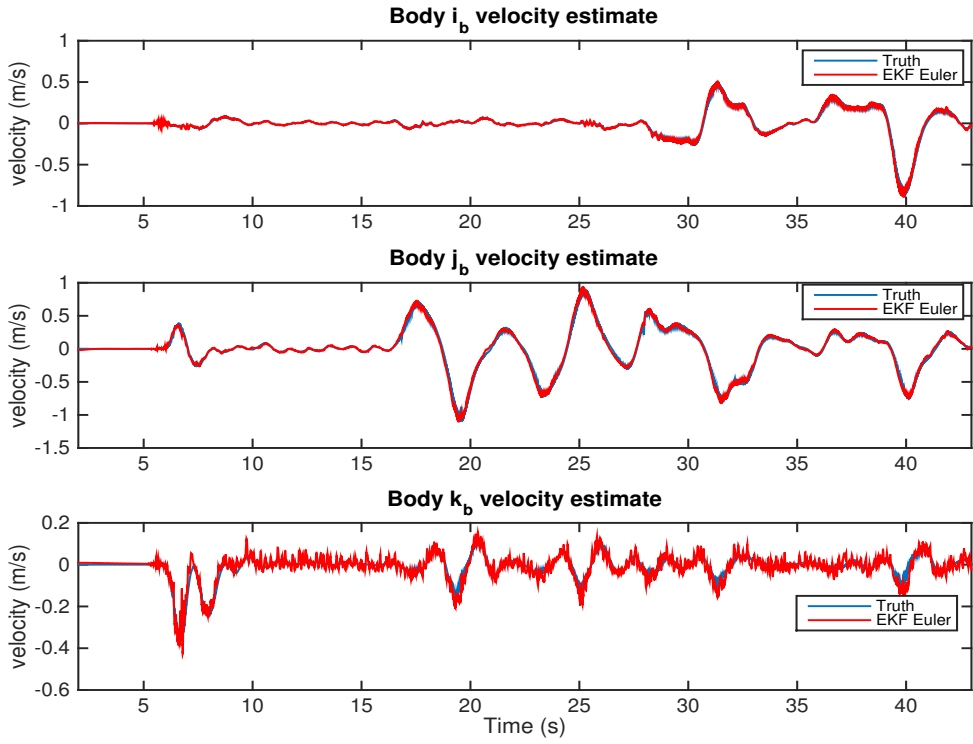


Figure 3.3: Estimates of velocities in body frame based on Euler model

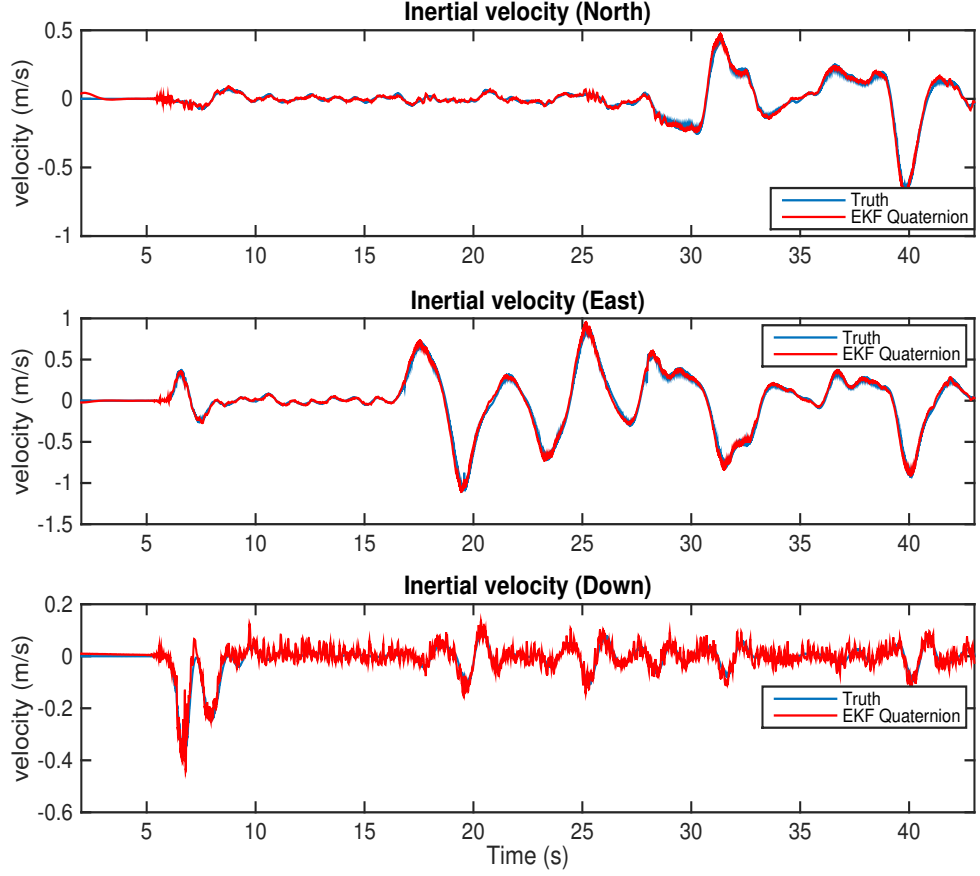


Figure 3.4: Estimates of velocities in inertial frame based on quaternion model

### 3.4.2 FULL STATE ESTIMATOR: CASE WITH ADDED BIASES

In this experiment, biases given by  $\beta = [-0.08, 0.12, -0.18]^T rad/s$  are injected into gyroscope measurements at time  $t = 30s$ , and  $\alpha = [0.5, -0.3, -0.6]^T m/s^2$  biases are added to the accelerometer measurements at time  $t = 45s$ . Figure 3.5 - 3.6 and Figure 3.9 - 3.10 shows the estimation results of inertial positions, Euler angles, gyroscope biases, and accelerometer biases with solid blue lines showing the true values of the parameters, red lines representing the Kalman Filter estimate based on Euler method, and the yellow lines representing the quaternion based Kalman Filter estimates. Additionally, Figure 3.7 shows the velocity estimate in the body frame generated by the Euler based implementation, whereas Figure 3.8 represents the inertial frame velocity estimate as a result of quaternion based Kalman Filter execution in the presence of biases. It can be seen that after the occurrence of biases, the quadrotor states shows

a slight divergence from the true value which is more prominent in the Euler angles, but the filter quickly estimates the biases and adjusts the gyroscope and accelerometer measurements. It should also be noted that the filter does a good job in estimating the state estimates even in the presence of accelerometer and gyroscope biases and keeps all the states estimates within acceptable limits.

Figure 3.9 and 3.10 shows the bias estimation of gyroscope and accelerometer. From these two figures, it can be seen that once the biases are added, Kalman Filter quickly estimates these biases to ensure good estimates of the other states. There are some biases already present in the measurement of gyroscope and accelerometer and are also estimated by the Kalman Filter along with the added biases.

It can be seen from Figure 3.5 - 3.8 that before the addition of biases, the state estimates closely resemble the true values. Whereas, after the occurrence of biases, there is a slight drift in the state estimation which is more prominent in the Euler angles than the other states. Since, the biases are estimated quickly, state estimation errors also reduce after the successful estimation of these biases.

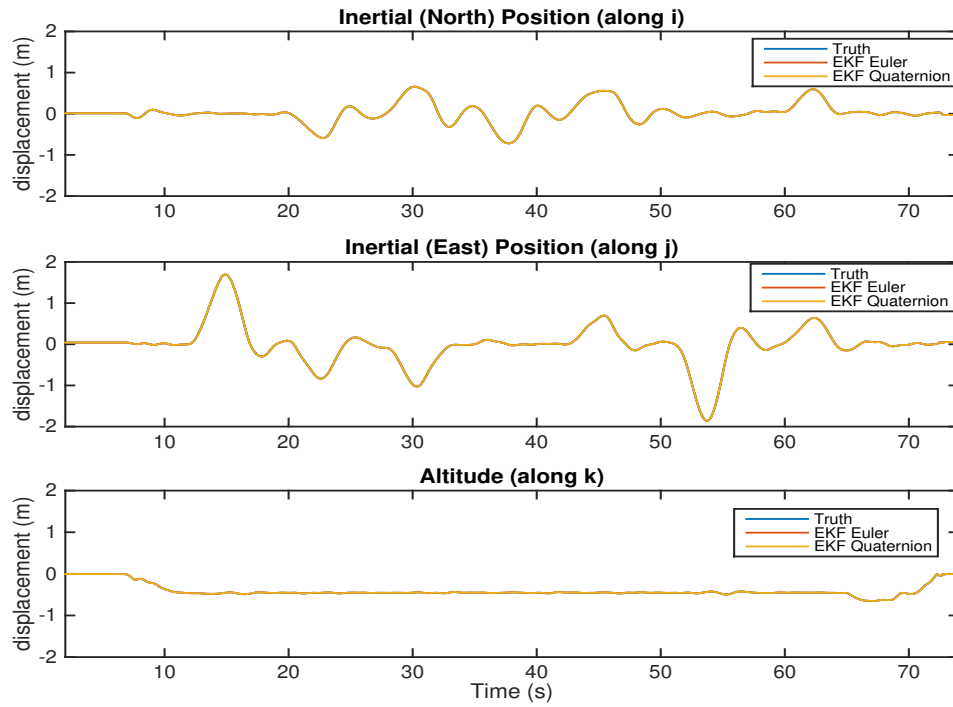


Figure 3.5: Estimates of inertial positions with added biases

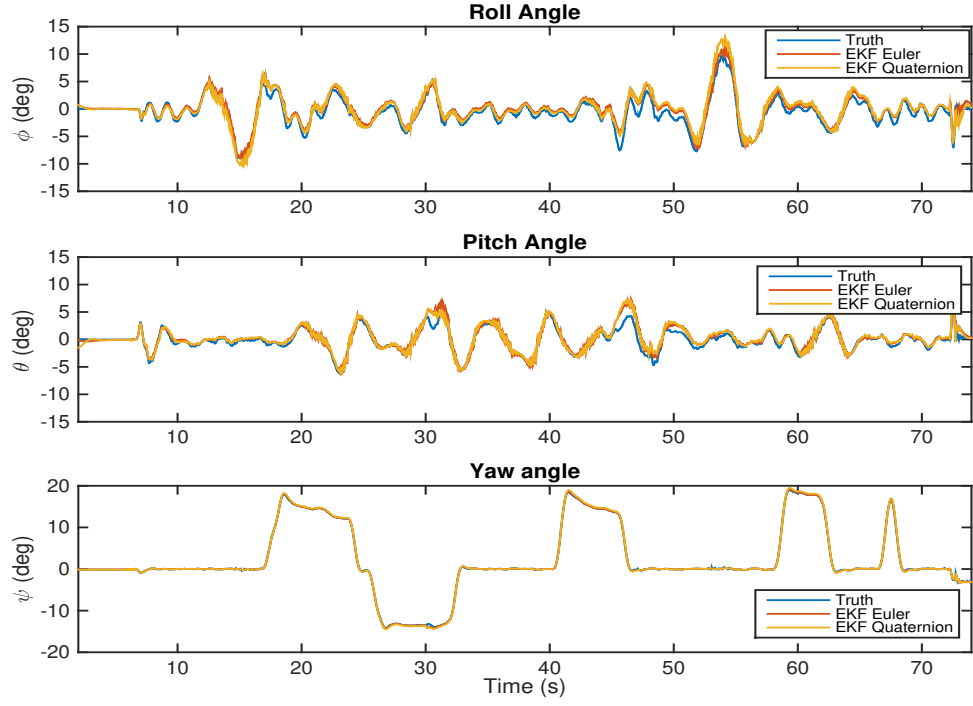


Figure 3.6: Estimates of Euler angles with added biases

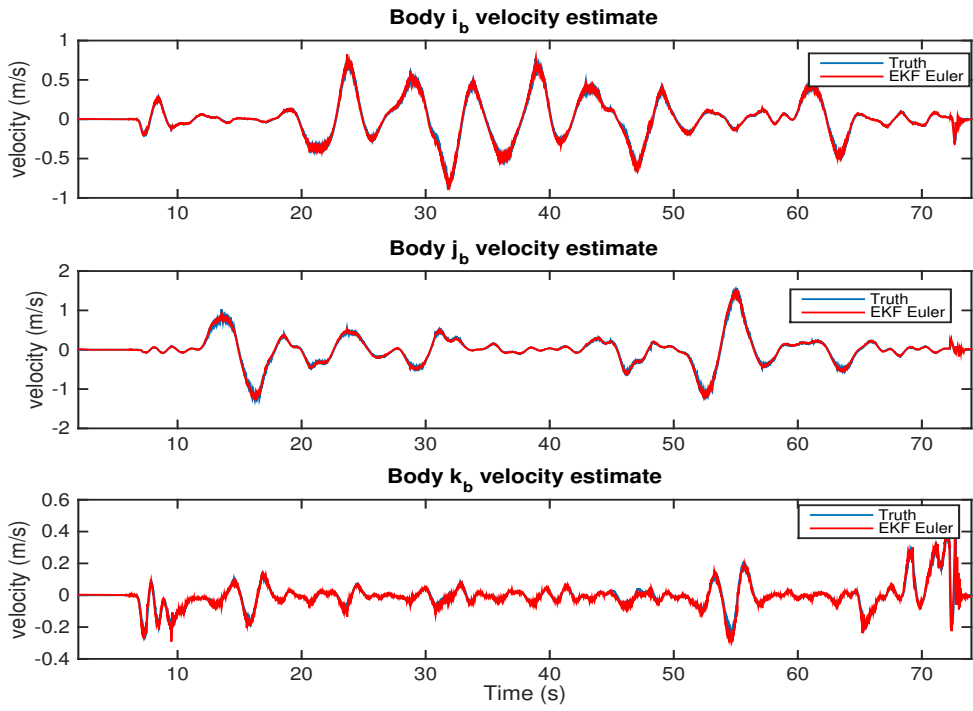


Figure 3.7: Estimates of velocities in body frame with added biases based on Euler model

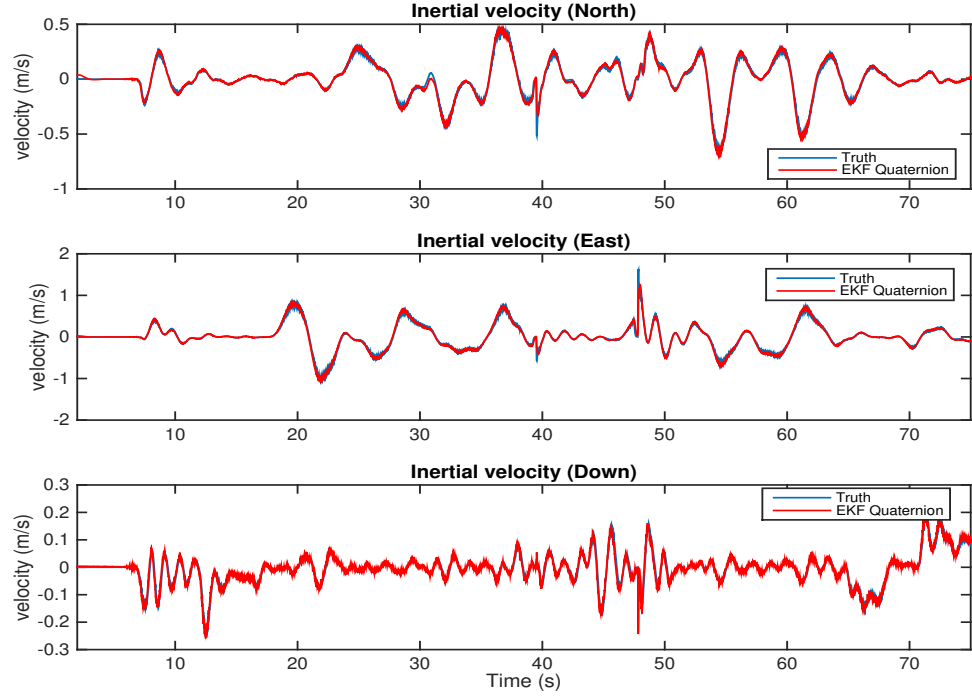


Figure 3.8: Estimates of velocities in inertial frame with added biases based on quaternion model

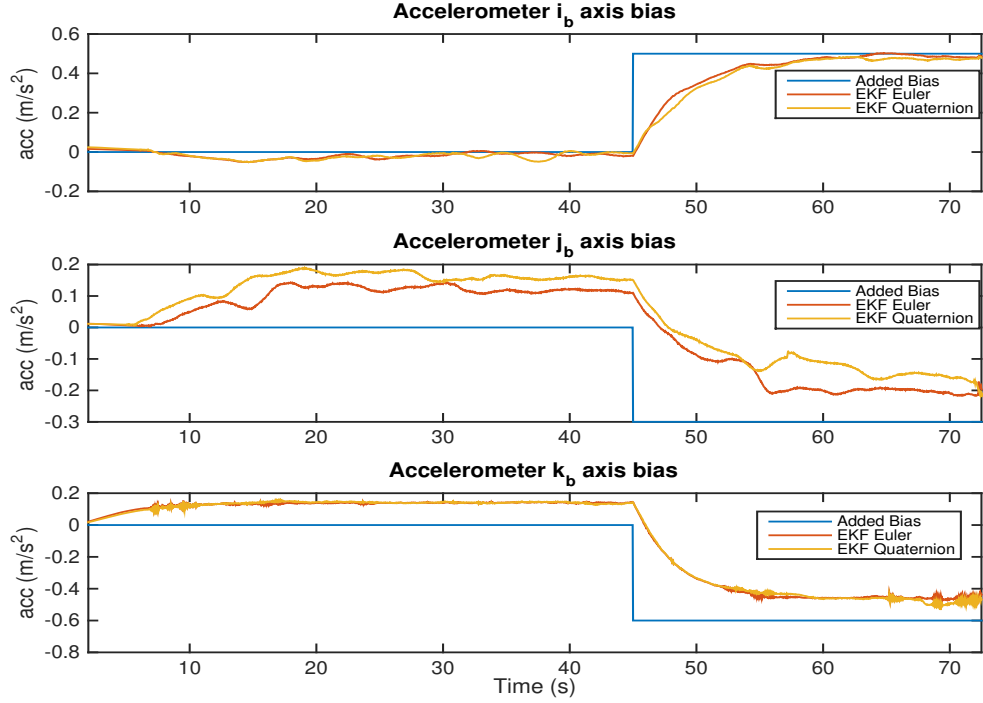


Figure 3.9: Estimates of accelerometer biases

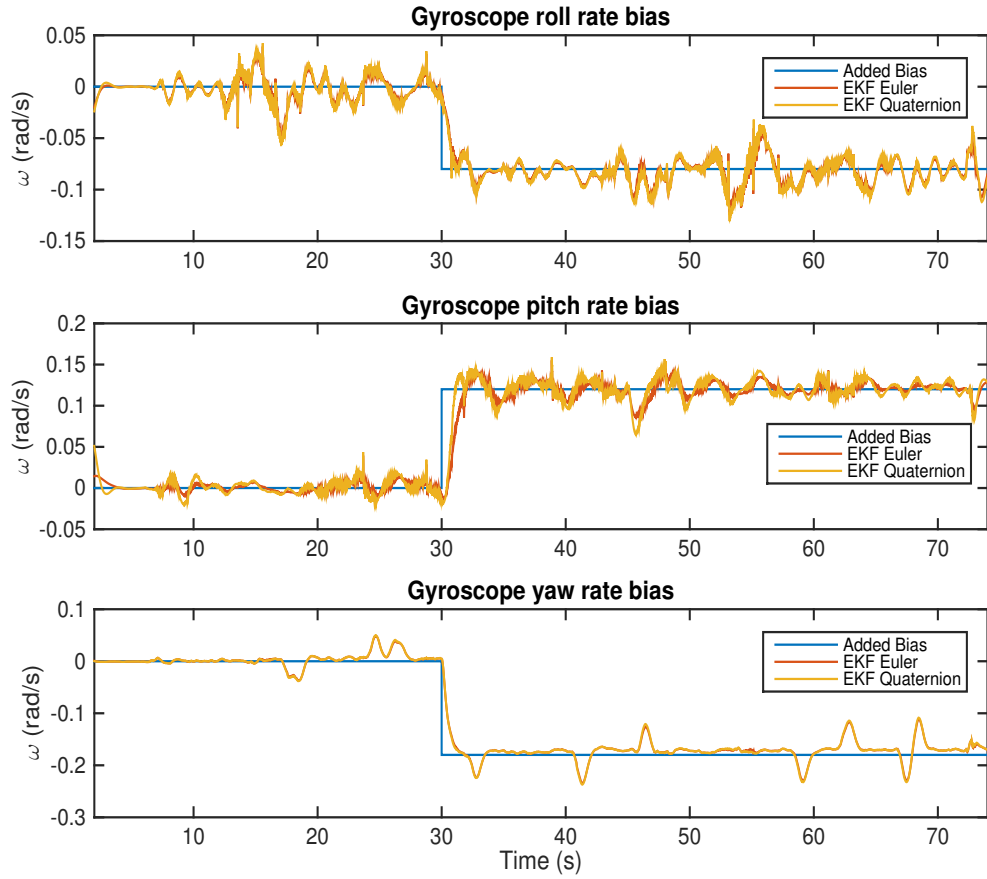


Figure 3.10: Estimates of gyroscope biases

### 3.4.3 ESTIMATING EULER ANGLES, ALTITUDE AND VERTICAL VELOCITY

In this situation, estimates generated by the EKF for Euler angles, altitude and vertical velocity will be discussed. Since, these estimates doesn't require the use of the Vicon camera system and rely only on the on-board sensors, so they can come very handy in situations when we lose the position measurements coming from the Vicon camera system(for indoor applications) or GPS (for outdoor purposes). Once we detect that the position measurements are lost, we can switch to this Kalman Filter to provide us enough information to successfully fly and land the quadrotor. The results shown in this experiment uses the Vicon signals for altitude and yaw angle measurements for testing purposes.

Figure 3.11 and 3.12 shows the estimates of altitude, inertial velocity along z, and Euler angles. Before the quadrotor takes off, ground calibration is done to reduce the constant biases present in the accelerometer and gyroscope measurements. These constant biases are then subtracted from the IMU measurements before being fed to the Kalman Filter.

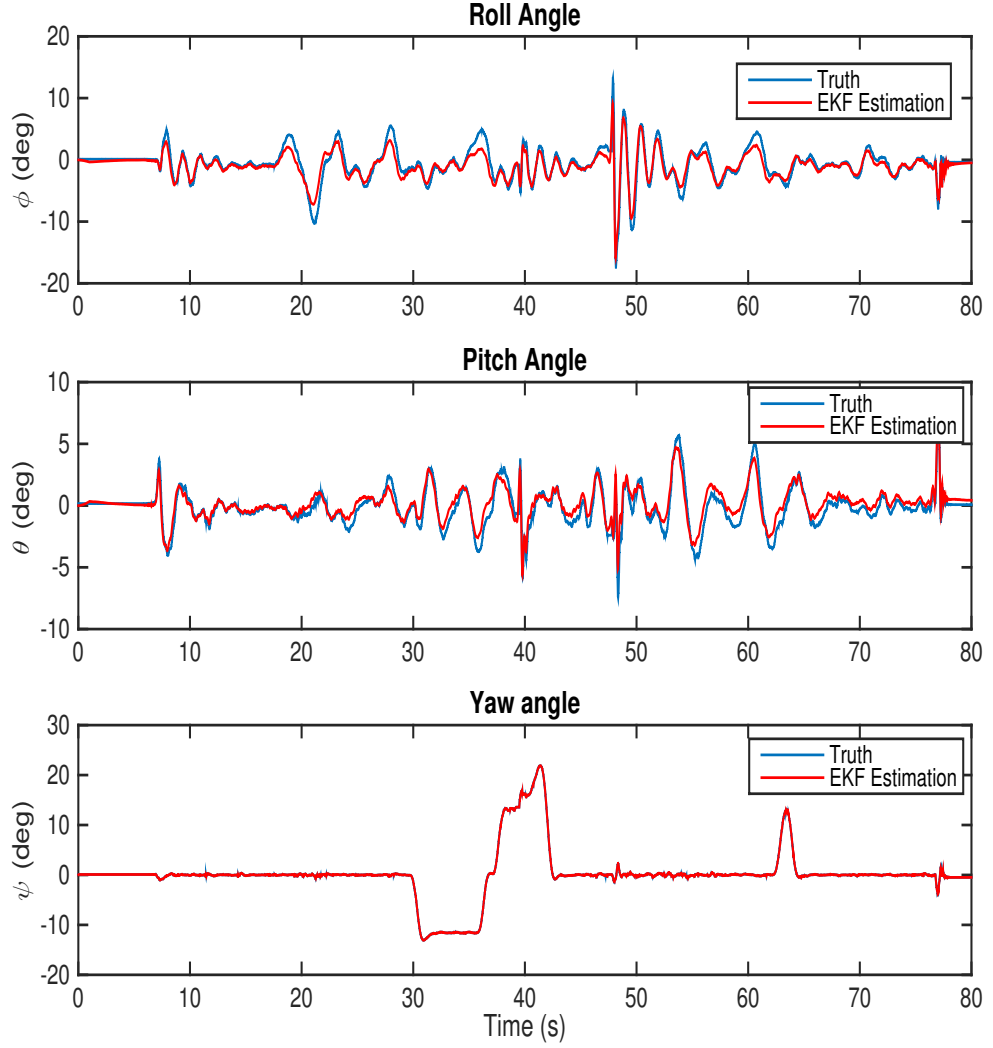


Figure 3.11: Estimates of Euler angles

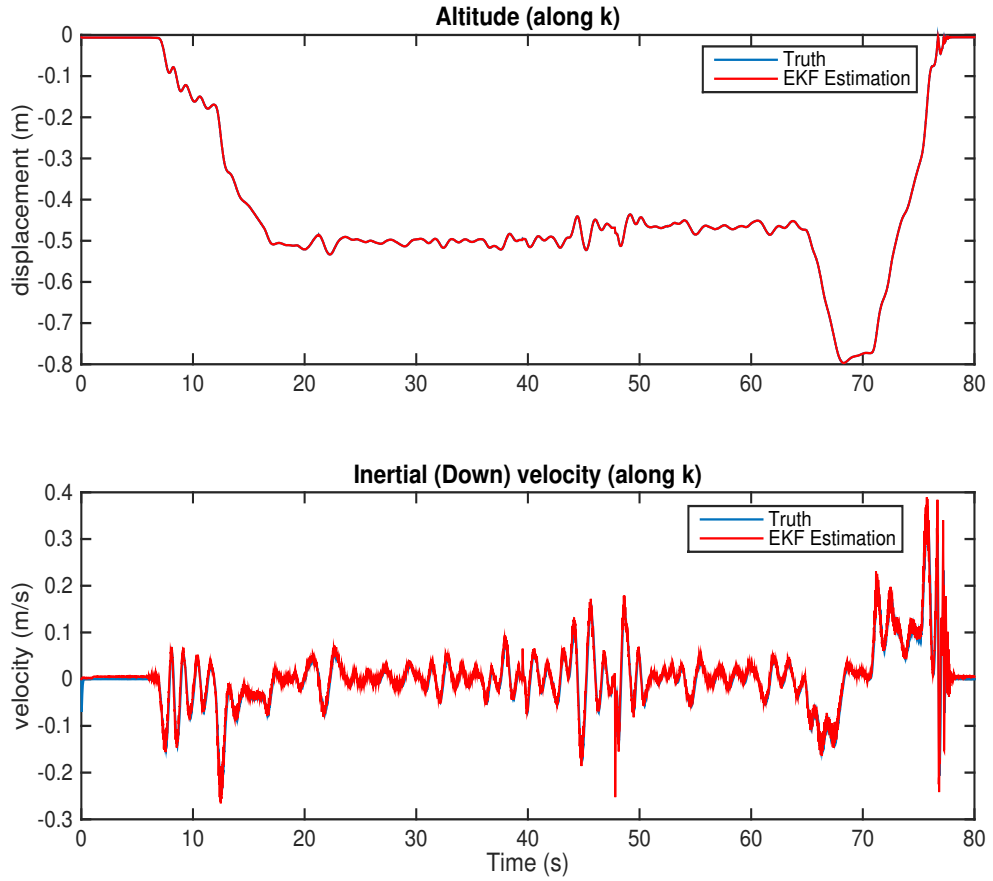


Figure 3.12: Estimates of altitude and vertical velocity

### 3.5 CONCLUSION

In this Chapter, the design of Extended Kalman Filter and its implementation on a real time system (quadrotor) has been discussed. Experimental results are generated as a result of the EKF implemented in two different ways to generate the following states.

- Positions, linear velocities, Euler angles, gyroscope biases, and accelerometer biases.
- Altitude, velocity associated with height and Euler angles.

For the first case, results are produced and discussed in the absence and presence of accelerometer and gyroscope biases. In the absence of IMU biases, results are shown for both the Euler and quaternion based Kalman Filter. The effect on the estimates due to the simultaneously added biases in accelerometer and gyroscope measurements at different times during real time flight has also been shown using the Euler and quaternion



based implementation of Kalman Filter. It can be seen that the results produced as a result of the EKF are quite accurate in the presence and absence of biases.

In the second case, states, i.e. altitude, Euler angles, and vertical velocity, used to control only the inner-loop are estimated. The experimental results for these estimated states are shown and discussed by using the Vicon signals for yaw angle and altitude measurements. This is done to test the working of the filter, but the main purpose of this EKF is to remove the dependence on GPS signals (for outdoor applications) and Vicon camera system (for indoor flights) by using only the on-board IMU, magnetometer and sonar sensors. So, after installing a sonar and a magnetometer sensor for altitude and yaw angle measurements, they will replace the Vicon system measurements. This makes real time quadrotor flight safer, especially during outdoor flights where GPS signal may get lost due to interference or jamming.

## 4. Linear Quadratic Regulator

### 4.1 INTRODUCTION

In this chapter, a Linear Quadratic Regulator (LQR) controller with integral action is developed for the quadrotor under consideration following a double loop control architecture. Position and yaw angle references are given directly during the implementation, whereas, roll and pitch angle references are extracted from the output of  $x$  and  $y$  position controllers (outer loop). (The feedback required during the process are taken from the Extended Kalman Filter described in Chapter 3.) In the event when position measurements are assumed to be lost, only the inner loop control is active during which  $x$  and  $y$  positions are not controlled and roll and pitch angle references are provided directly. In order to design the feedback gain for the LQR controller, the nonlinear quadrotor model is linearized. The designed LQR controller for the required states is implemented and demonstrated using real time flight tests performed on a quadrotor.

### 4.2 GENERAL OVERVIEW OF THE LINEAR QUADRATIC REGULATOR

Linear Quadratic Regulator control is an algorithm developed for systems with constraints on inputs and outputs to determine the state feedback control law which minimizes a quadratic performance criterion [29]. The idea of designing a feedback controller which minimizes the integral of square of tracking error is first proposed by Hall [30] and Wiener [31], and further developed by Newton, Gould and Kaiser [32]. Immediately after its appearance, Linear Quadratic problem became a part of many influential books [33–36].

Lets assume that the linear state space model for the quadrotor is given below:

$$\dot{x} = Ax + Bu, \quad (4.1)$$

where  $A \in \mathbb{R}^{n \times n}$  is the state matrix,  $B \in \mathbb{R}^{n \times m}$  is defined as the input matrix,  $x$  is the state vector containing  $n$  number of states, and  $u$  is the input vector with  $m$  inputs. The pair  $(A, B)$  must be controllable in order to design the LQR controller.

An LQR controller can be designed to give us an optimal controller which ensures maximum performance and keeps the control signals within the physical constraints. The main objective is to find a feedback gain  $K$ , resulting in optimal control action,

which minimizes the cost function given below:

$$J = \int_0^\infty (x^T Q x + u^T R u) dt \quad (4.2)$$

where  $R \in \mathbb{R}^{n \times n}$  and  $Q \in \mathbb{R}^{m \times m}$  are positive definite symmetric weighing matrices associated with the inputs and the states. Large value of a diagonal element in  $Q$  matrix means that the state corresponding to that element needs to be small to keep  $J$  small. Therefore, larger elements of  $Q$  causes faster convergence of the states. Similarly, large value of  $R$  results in smaller control inputs and hence larger values of the states.

The control input resulting from the feedback gain is given below:

$$u = -Kx, \quad (4.3)$$

where  $K$  is the feedback gain and it is determined using the following expression:

$$K = R^{-1} B^T P, \quad (4.4)$$

where  $P$  is a positive definite symmetric matrix and a solution of the Riccati's algebraic equation:

$$A^T P + P A + Q - P B R^{-1} B^T P = 0. \quad (4.5)$$

The location of the closed-loop poles are changed by varying  $Q$  and  $R$  matrices, thereby changing the performance of the system. The  $Q$  and  $R$  matrices can be initially chosen based on Bryson's rule. The resulting  $Q$  and  $R$  matrices are diagonal matrices with elements as shown below:

$$Q_i = \frac{1}{x_{i_m}^2}, \quad R_j = \frac{1}{u_{j_m}^2}, \quad (4.6)$$

where  $x_{i_m}$  is the maximum desired value for the  $i^{th}$  state, and  $u_{j_m}$  is the maximum desired value for the  $j^{th}$  input and it can be chosen to be the saturation value of the control signal.

Response of the experimental quadrotor system is observed for the initial values of the weighing matrices given by (4.6). These values are varied until the desired response

is achieved.

### 4.3 CONTROL SYSTEM DESIGN

#### 4.3.1 CONTROL ARCHITECTURE

In order to achieve full control of a quadrotor, two feedback control loops are implemented. The inner loop controls the altitude and attitude, whereas the outer loop controls the x and y translational positions. The reference signals for the roll and pitch angles are generated by the x and y controller outputs. All the other reference signals are provided by the control station directly. Figure 4.1 shows the implemented control structure using the LQR technique with the feedback/measurements coming from the EKF discussed in the previous chapter.

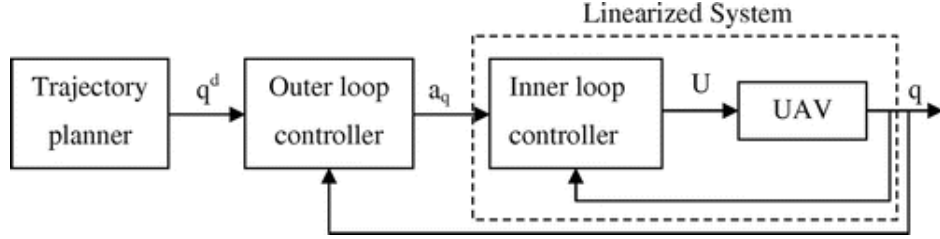


Figure 4.1: Architecture of the control loops

In order to solve this problem, the nonlinear quadrotor model is first linearized, and then the state space model is divided to design LQR controllers for x and y translational positions, attitude, and altitude to fully control the quadrotor. The general block diagram of an LQR controller with integral action is given below:

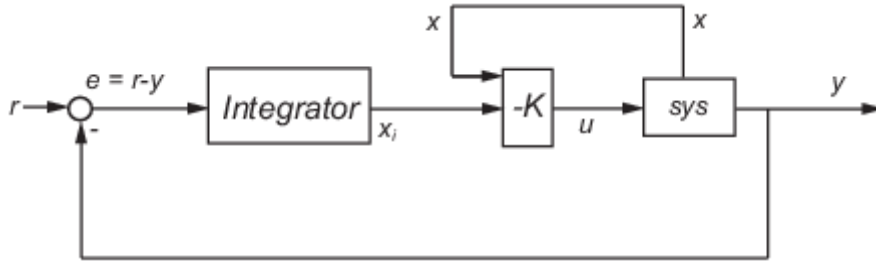


Figure 4.2: Block diagram of LQR control with integral action

Once the nonlinear system model is linearized, gain K along with the integral gain can be designed for the Euler angles, altitude and translational positions.

Figure 4.3 - 4.4 shows the experimental implementation of the LQR controller with the EKF estimates used as feedback signals. The inner and outer loop control blocks contains the LQR control structure shown in Figure 4.2. For the case when the position measurements are available and all the states are estimated, the control architecture shown by Figure 4.3 is implemented. Whereas, when the position measurements are lost, the experimental model switches to the second EKF providing enough feedback signals to control the inner loop parameters represented by Figure 4.4.

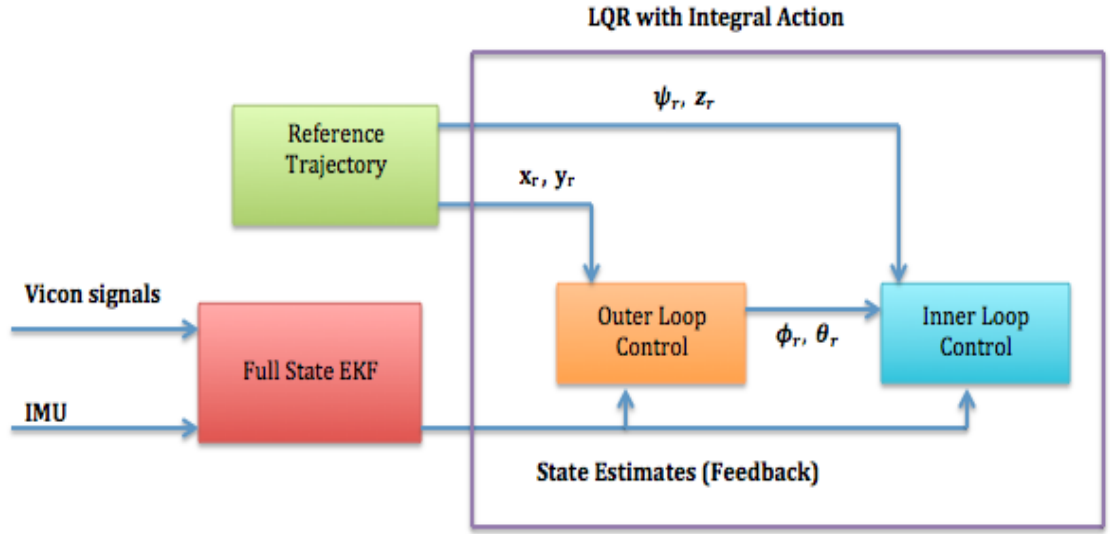


Figure 4.3: Block diagram of the experimental setup when position measurements are available

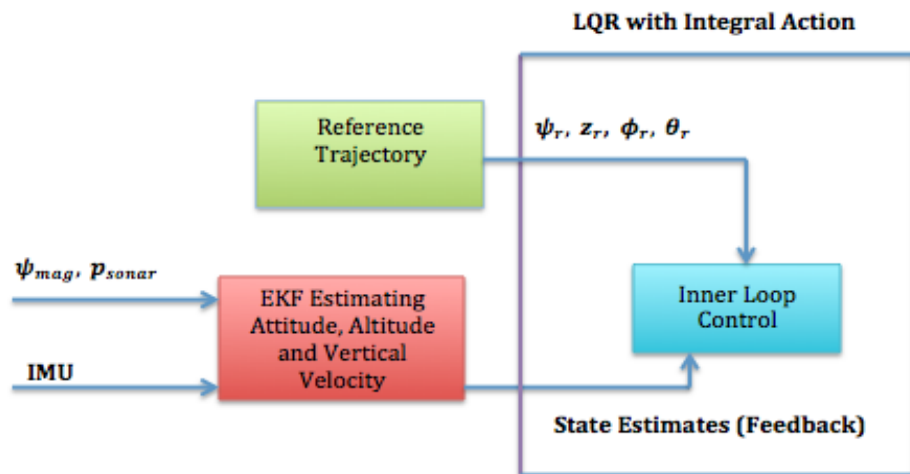


Figure 4.4: Block diagram of the experimental setup when position measurements are not available

### 4.3.2 CONTROL LAW DESIGN SPECIFICATIONS

#### 1. Roll and Pitch Angles

The dynamics of the roll and pitch angles resemble a lot due to which similar controller gains can be used for both of them. The controller is designed to meet the following specifications for the roll and pitch angles.

- Overshoot  $\leq 20\%$
- Rise Time  $\leq 0.5$  sec

#### 2. Yaw angle and Altitude

Design specifications chosen for yaw angle are:

- Overshoot  $\leq 5\%$
- Rise Time  $\leq 0.7$  sec

Similarly, the following design specifications for altitude are:

- Overshoot  $\leq 15\%$
- Rise Time  $\leq 1.5$  sec

#### 3. x and y Positions

The outer loop parameters, i.e., x and y positions are dependent on the roll and pitch angles. Therefore, the response of the roll and pitch angles must be quicker as compared to the x and y positions. Keeping that in mind, the required design specifications for the x and y positions are set to be the following.

- Overshoot  $\leq 10\%$
- Rise Time  $\leq 2$  sec

### 4.3.3 LINEARIZATION

The state space model consists of the following states:

$$x = [p_{x_i} \ p_{y_i} \ p_{z_i} \ v_x \ v_y \ v_z \ \phi \ \theta \ \psi \ p \ q \ r]^T, \quad (4.7)$$

where  $p_{x_i}$ ,  $p_{y_i}$  and  $p_{z_i}$  are the inertial positions,  $v_x$ ,  $v_y$  and  $v_z$  are the velocities expressed in the inertial frame,  $\phi$ ,  $\theta$  and  $\psi$  are the roll, pitch and yaw angles, and  $p, q$  and  $r$  are

the angular rates expressed in body frame of the quadrotor. In order to linearize our model to the form given by (4.1), the following expressions are used to compute  $A$  and  $B$  matrices:

$$A = \left[ \frac{\partial f}{\partial x} \right]_{(x_e, u_e)}, \quad B = \left[ \frac{\partial f}{\partial u} \right]_{(x_e, u_e)} \quad (4.8)$$

where  $x_e$  and  $u_e$  are the states and inputs at the equilibrium point. The equilibrium point is taken to be the point when the quadrotor is in hover position.

After solving the Jacobian matrices given by (4.8) and plugging in the equilibrium states and inputs, we get the following  $A$  and  $B$  matrices:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.235 & 0 & 0 & 0.1827 & -9.81 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.235 & 0 & 9.808 & 0.1827 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.235 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -0.111 & -5.98 & -0.432 & 0 & 0 & 0 & -3.71 & -0.569 & 0 & 0 \\ 0 & 0 & 0 & 4.453 & -0.083 & -1.032 & 0 & 0 & 0 & -0.149 & -3.842 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.454 & 0 & 0 & 0 & 0.052 & -0.18 & -0.614 & 0 \end{bmatrix} \quad (4.9)$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -0.784 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 45.872 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 45.045 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 32.68 & 0 \end{bmatrix}^T. \quad (4.10)$$

The linear state space model can be divided to make it suitable for implementing LQR technique to control Euler angles, altitude and translational positions.

#### 4.3.4 ATTITUDE CONTROL

In this case, the state space model only considers the following states:

$$x_e = [\phi \ \theta \ \psi \ p \ q \ r \ v_x \ v_y]^T. \quad (4.11)$$

The state and input matrices corresponding to the attitude control are:

$$A_e = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -3.71 & -0.569 & 0 & -0.111 & -5.98 \\ 0 & 0 & 0 & -0.149 & -3.842 & 0 & 4.453 & -0.083 \\ 0 & 0 & 0 & 0.052 & -0.18 & -0.614 & 0 & 0 \\ 0.1827 & -9.81 & 0 & 0 & 0 & 0 & -0.235 & 0 \\ 9.808 & 0.1827 & 0 & 0 & 0 & 0 & 0 & -0.235 \end{bmatrix}, \quad (4.12)$$

$$B_e = \begin{bmatrix} 0 & 0 & 0 & 45.872 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 45.045 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 32.68 & 0 & 0 \end{bmatrix}^T.$$

Similarly, the output vector  $y_e$  and control signals  $u_e$  in this case are:

$$y_e = [\phi \ \theta \ \psi]^T, \quad u_e = [\tau_\phi \ \tau_\theta \ \tau_\psi]^T, \quad (4.13)$$

where  $\tau_\phi$ ,  $\tau_\theta$ , and  $\tau_\psi$  are the rolling torque, pitching torque, and the yawing torque, respectively.

The  $Q$  and  $R$  matrices selected for the control of attitude are given below:

$$Q_e = \text{diag}([1 \ 1 \ 1 \ 1.5 \ 1.5 \ 1.5 \ 0.5 \ 0.5 \ 800 \ 800 \ 900]), \quad R_e = \text{diag}([16 \ 16 \ 25]). \quad (4.14)$$

The gain matrix  $K$  and the integral gains for roll, pitch, and yaw are calculated to be



the following:

$$K_e = \begin{bmatrix} 2.58 & 0.03 & 0.008 & 0.38 & -0.007 & 0.001 & -0.002 & -0.1 \\ -0.03 & 2.59 & -0.03 & -0.006 & 0.38 & -0.004 & 0.073 & -0.002 \\ -0.007 & 0.02 & 2.33 & 0.001 & -0.002 & 0.43 & 0 & 0 \end{bmatrix}, \quad (4.15)$$

$$\begin{bmatrix} K_{i_\phi} & K_{i_\theta} & K_{i_\psi} \end{bmatrix} = \begin{bmatrix} -7 & -7 & -6 \end{bmatrix}.$$

Note that the off-axis quantities are very close to zero and can possibly be neglected without affecting the control performance, but they are still considered during implementation.

Table 4.1 shows the closed loop poles, damping ratio, and the natural frequency resulting from the feedback gain matrix for the attitude control shown in (4.15). It can be seen that all the Eigen values are in the left hand plane, and hence ensuring stability.

Poles	Damping Ratio ( $\zeta$ )	Natural Frequency ( $\omega_n$ )
-10.4 + 2.25i	0.977	10.6
-10.4 - 2.25i	0.977	10.6
-10.6 + 2.45i	0.974	10.8
-10.6 - 2.45i	0.974	10.8
-7.28 + 4.55i	0.848	8.58
-7.28 - 4.55i	0.848	8.58
-0.358	1	0.358
-0.339	1	0.339

Table 4.1: Eigen values, damping ratio, and natural frequency

#### 4.3.5 ALTITUDE CONTROL

For the altitude control, we neglect all states except the following.

$$x_{alt} = [p_{x_z} \ v_z]^T. \quad (4.16)$$

which results in the approximate model described by:

$$A_{alt} = \begin{bmatrix} 0 & 1 \\ 0 & -0.235 \end{bmatrix}, \quad B_{alt} = \begin{bmatrix} 0 & -0.784 \end{bmatrix}^T. \quad (4.17)$$

In this case, the output and input are the following:

$$y_{alt} = p_{z_i}, \quad u_{alt} = F, \quad (4.18)$$

where  $F$  is the thrust force.

Using the weighting matrices:

$$Q_{alt} = \text{diag}([0.2 \quad 0.1 \quad 3]), \quad R_{alt} = 0.2, \quad (4.19)$$

we generate the following feedback and integral gains:

$$K_{alt} = \begin{bmatrix} -5.54 & -3.54 \end{bmatrix}, \quad K_{i_{alt}} = 3.9. \quad (4.20)$$

Note that the negative signs in the gain matrix is because the z-axis is pointed downwards. In order to move the quadrotor in the upward direction away from the ground, a negative altitude reference must be given.

The designed feedback gain results in stable closed loop poles and are shown below in Table 4.2.

Poles	Damping Ratio ( $\zeta$ )	Natural Frequency ( $\omega_n$ )
$-1.63 + 1.57i$	0.72	2.26
$-1.63 - 1.57i$	0.72	2.26

Table 4.2: Eigen values, damping ratio, and natural frequency

#### 4.3.6 X AND Y POSITION CONTROL

The states involved in the control of x and y positions are:

$$x = [p_{x_i} \quad p_{y_i} \quad v_x \quad v_y]^T. \quad (4.21)$$

Therefore, the corresponding state and input matrices for designing x-y position controller are:

$$A_{xy} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -0.235 & 0 \\ 0 & 0 & 0 & -0.235 \end{bmatrix}, \quad B_{xy} = \begin{bmatrix} 0 & 0 & 0.1827 & 9.808 \\ 0 & 0 & -9.81 & 0.1827 \end{bmatrix}^T. \quad (4.22)$$

Similarly, the output vector  $y_p$  and control signals  $u_p$  are:

$$y_p = \begin{bmatrix} p_{x_i} & p_{y_i} \end{bmatrix}^T, \quad u_p = \begin{bmatrix} \phi & \theta \end{bmatrix}^T. \quad (4.23)$$

The following diagonal weighting matrices  $Q$  and  $R$  are chosen:

$$Q_{xy} = \text{diag}([0.01 \quad 0.01 \quad 0.001 \quad 0.001 \quad 0.12 \quad 0.12]), \quad R_{xy} = \text{diag}([2.5 \quad 2.5]). \quad (4.24)$$

Now, using (4.4) and (4.5), the following feedback gain matrix is obtained.

$$K_{xy} = \begin{bmatrix} 0.007 & 0.34 & 0.005 & 0.24 \\ -0.34 & 0.007 & -0.24 & 0.005 \end{bmatrix}, \quad \begin{bmatrix} K_{i_x} & K_{i_y} \end{bmatrix} = \begin{bmatrix} 0.22 & -0.22 \end{bmatrix}, \quad (4.25)$$

where  $K_{xy}$  represents the feedback gain matrix, and  $K_{i_x}$  and  $K_{i_y}$  are the integral gains for x and y position control respectively.

It can be observed from (4.25) that the off-axis terms in gain matrix for both x and y position controls are very small and neglecting them wouldn't do any harm while implementation.

The closed loop poles as a result of the designed gains are shown in Table 4.3. It is noted that all the Eigen values are negative, which corresponds to a stable system.

Poles	Damping Ratio ( $\zeta$ )	Natural Frequency ( $\omega_n$ )
-1.32 + 1.3i	0.712	1.85
-1.32 - 1.3i	0.712	1.85
-1.32 + 1.3i	0.712	1.85
-1.32 - 1.3i	0.712	1.85

Table 4.3: Eigen values, damping ratio, and natural frequency

## 4.4 SIMULATION RESULTS

In this section, simulation results for the designed LQR controller with integral action are described. In order to evaluate the performance of the controller, a Simulink model is developed to implement the LQR controller and the nonlinear system dynamics described in Chapter 2. The feedback for the states are generated by integrating the nonlinear quadrotor system dynamics. The implemented Simulink model is divided into three parts, i.e., altitude control, attitude control, and x and y position control. Figure 4.5 shows the implemented Simulink model for the designed LQR controller to show the effectiveness of the algorithm. The red, green, and yellow colored subsystems/boxes represent the altitude, attitude, and x and y position controllers, respectively. The body of these subsystems contains the implemented control loop shown in Figure 4.2 and the nonlinear system dynamics described in Chapter 2. Note that based on the control loop architecture shown in Figure 4.1, the roll and pitch angle references are generated by the output of the x and y position controllers.

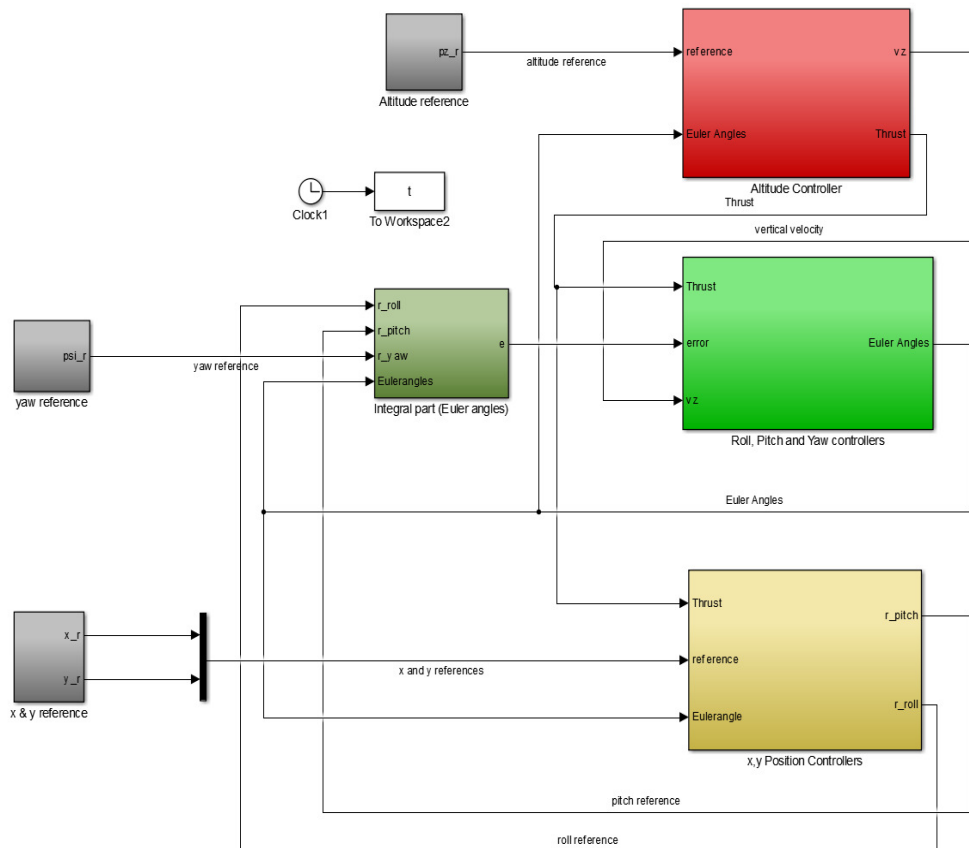


Figure 4.5: Simulink Model for the LQR controller

In order to verify the tracking performance of the designed LQR controller, step responses of the states are observed using the Simulation model for both the inner and the double loop control. Figure 4.6 - 4.7 shows the step response of Euler angles and altitude when only the inner loop control is established. Figure 4.8 - 4.10 shows the response of x and y positions, Euler angles, and altitude in response to the step commands applied to the x-y positions and altitude. The tracking performance is acceptable and within the design specifications.

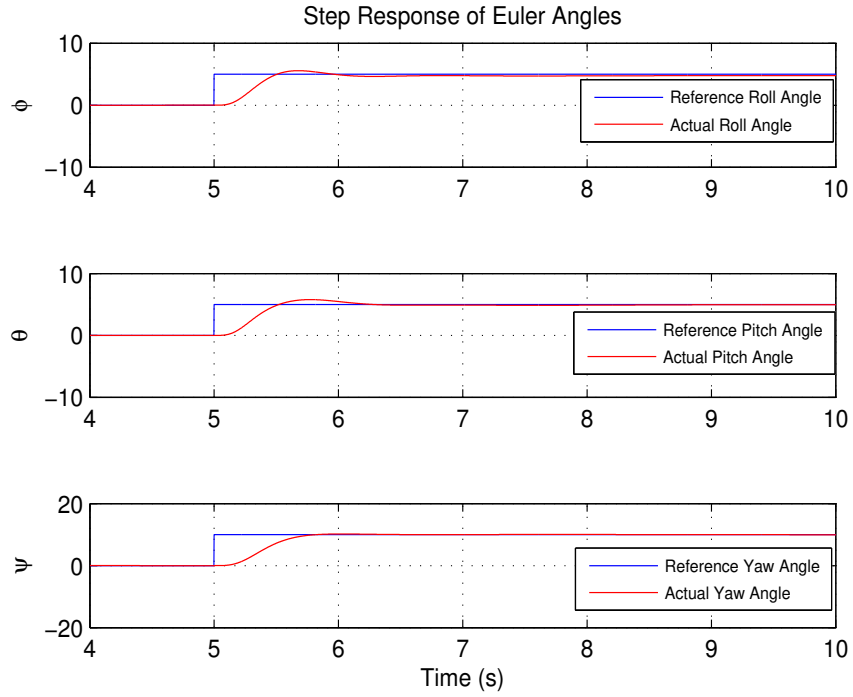


Figure 4.6: Step Response of Euler angles (Simulation Result)

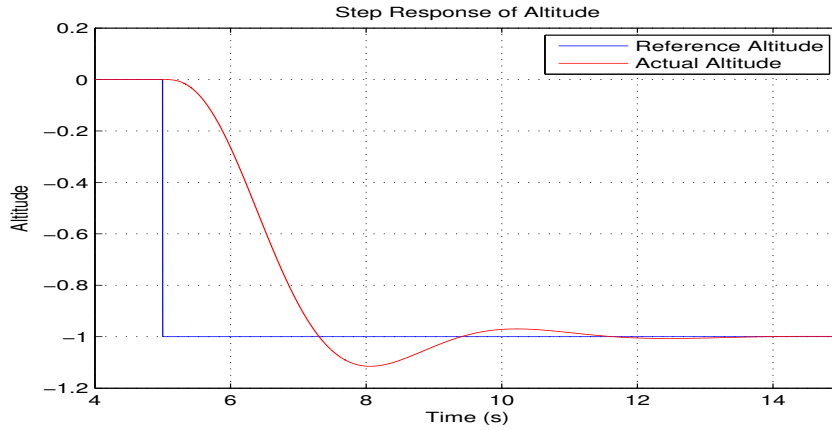


Figure 4.7: Step response of altitude (Simulation Result)

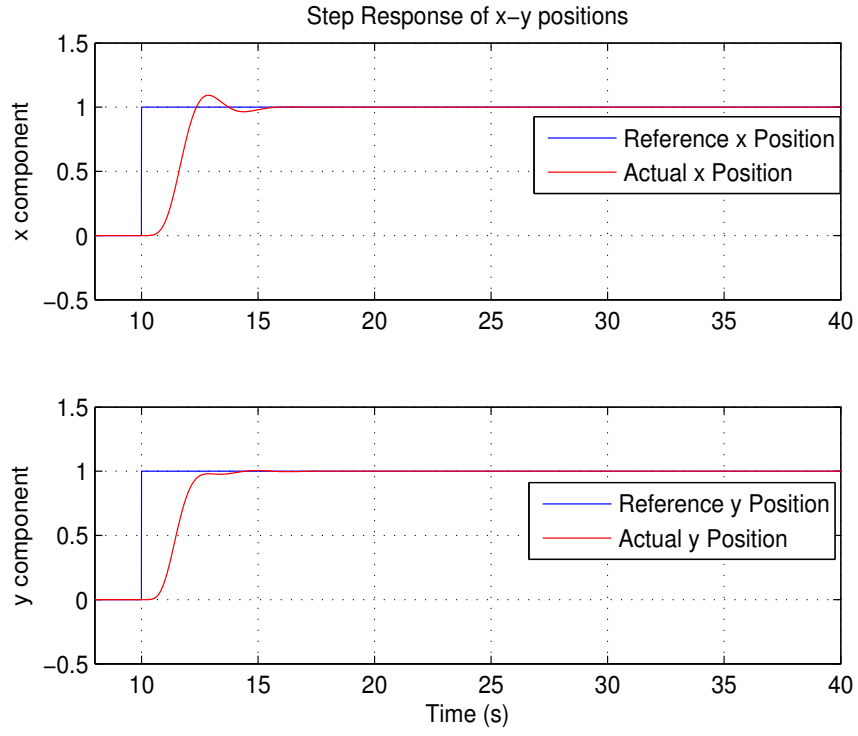


Figure 4.8: Step Response of x-y positions (Simulation Result)

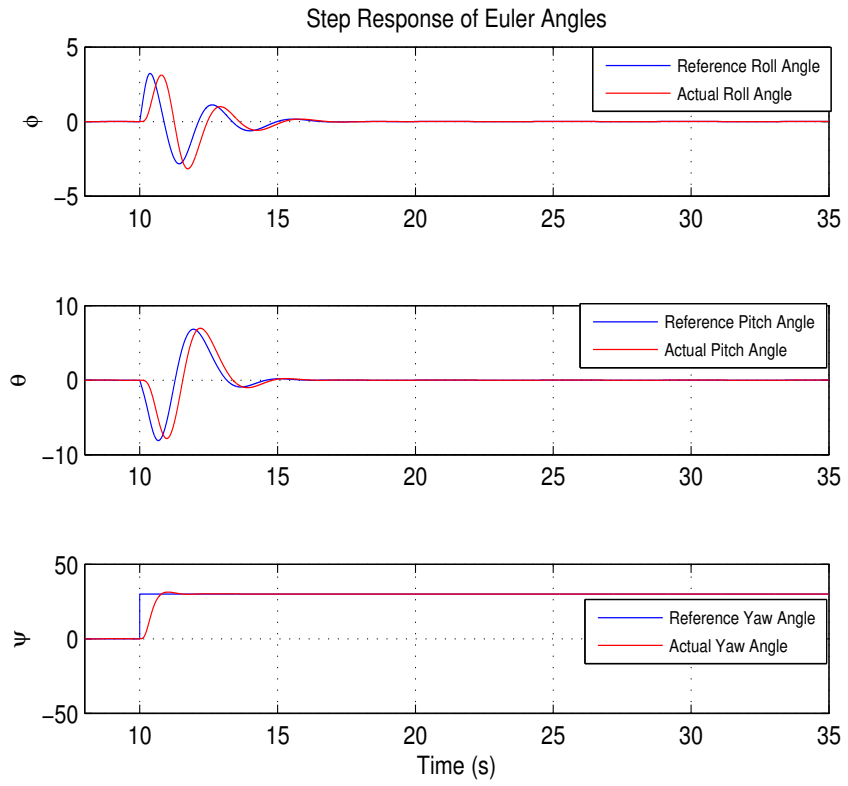


Figure 4.9: Step Response of Euler angles (Simulation Result)

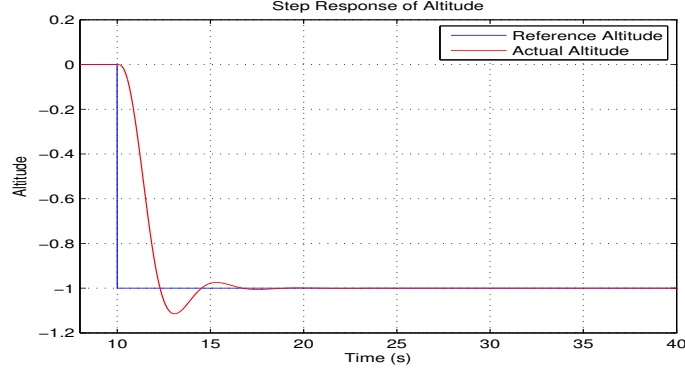


Figure 4.10: Step response of altitude (Simulation Result)

To further validate the effectiveness of the designed LQR controller, the tracking performance is observed in response to the following reference trajectory:

- Quadrotor is commanded to reach an altitude of one meter away from the origin of the inertial frame, i.e.,

$$alt_r = -1, \quad (4.26)$$

where  $alt_r$  is the altitude reference signal.

- After the quadrotor reaches the reference altitude, the following x and y position reference commands are generated.

$$x_r = 0.8\sin(0.5t) \quad (4.27)$$

$$y_r = 0.8\cos(0.5t), \quad (4.28)$$

where  $x_r$  and  $y_r$  are the reference signals for x and y positions, respectively.

Figure 4.11 - 4.13 shows the tracking performance of x-y positions, Euler angles and altitude, generated by the implemented LQR controller. It can be observed from these figures that all the states are tracking the reference signals nicely and the quadrotor is able to follow the Simulink generated reference trajectory. Although there seems to be a little lag in the tracking performance of the x and y position controller, the overall performance looks quite promising. This could be due to the inner/outer loop response time, and generally it is natural to have some lag between actual and commanded position. The Euler angles are able to track the reference signals accurately without any considerable lag which is seen in the case of the translational positions.

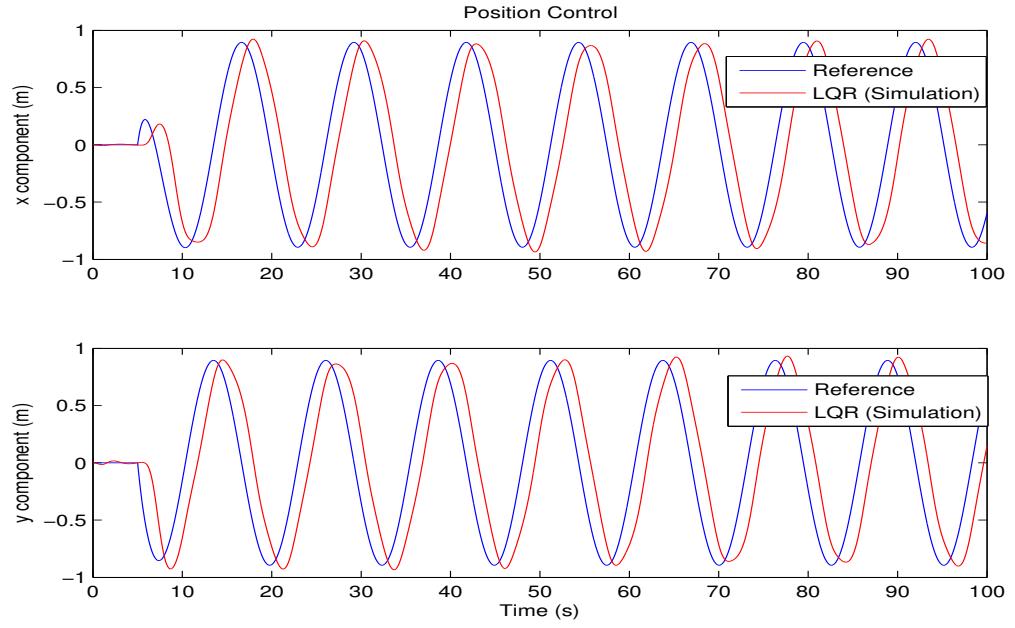


Figure 4.11: Position tracking using LQR technique (Simulation Result)

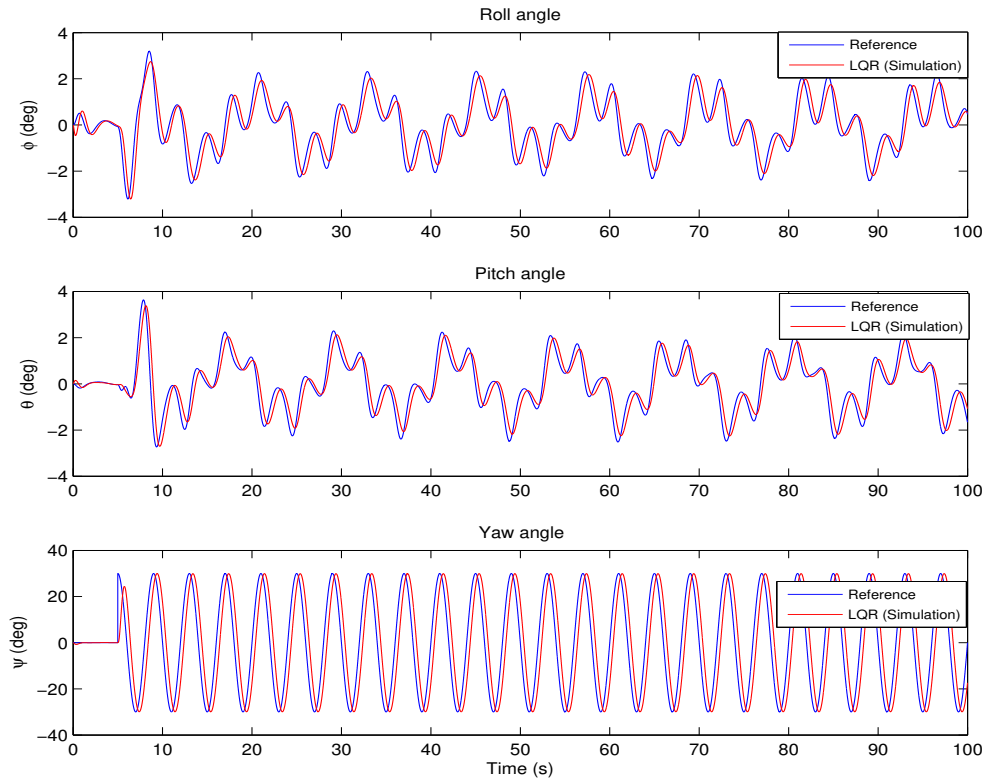


Figure 4.12: Roll, Pitch, and Yaw angle tracking using LQR technique (Simulation Result)



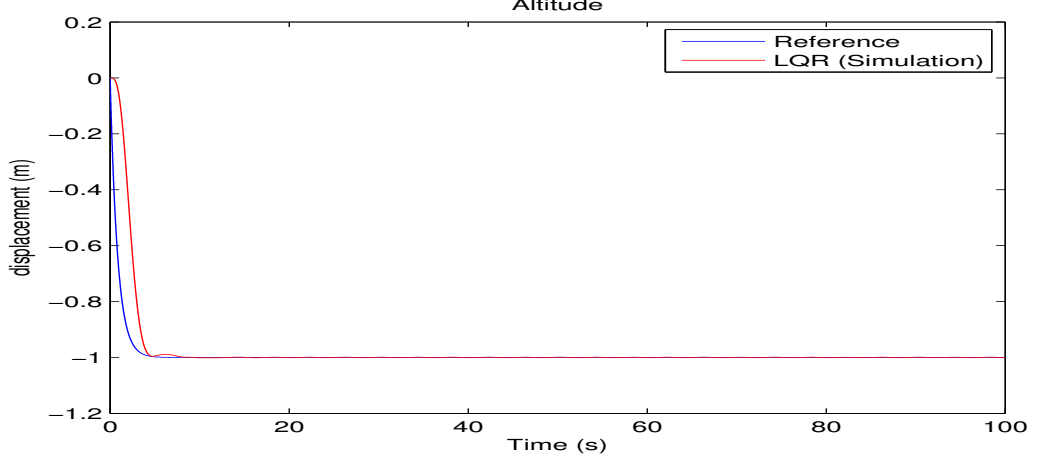


Figure 4.13: Altitude tracking using LQR technique (Simulation Result)

#### 4.5 EXPERIMENTAL RESULTS

In this section, real-time experimental results using an indoor quadrotor flight test environment are presented to show the effectiveness of the implemented LQR controller. During the implementation, the Kalman Filter described in Chapter 3 is used to provide the state information and no direct sensor values are used for feedback. Three different models are implemented, i.e. for attitude, altitude, and x-y position control. All of these parameters are controlled simultaneously using the feedback and integral gains calculated while designing the LQR controllers. In case of the outer loop control, roll and pitch angle references are generated as the output of the x-y controller. Whereas when only the inner loop control is established, roll and pitch angle references are given directly from the control station. Figure 4.15 - 4.16 shows the tracking performance of attitude, altitude, and x-y positions, respectively. Tracking performance of the Euler angles looks very promising, whereas, position tracking have more lag (as compared with Euler angle tracking) between the applied reference and the actual position. Nevertheless, the overall tracking performance of the implemented LQR controller is quite good. Figure 4.17 - 4.19 shows the adjusted body rates, inertial velocities, and the actuator control signals. It is worth noting that the rates are within the allowable range and the control signals are not saturating during the entire real-time flight, which is a requirement for a controller designed for practical systems.

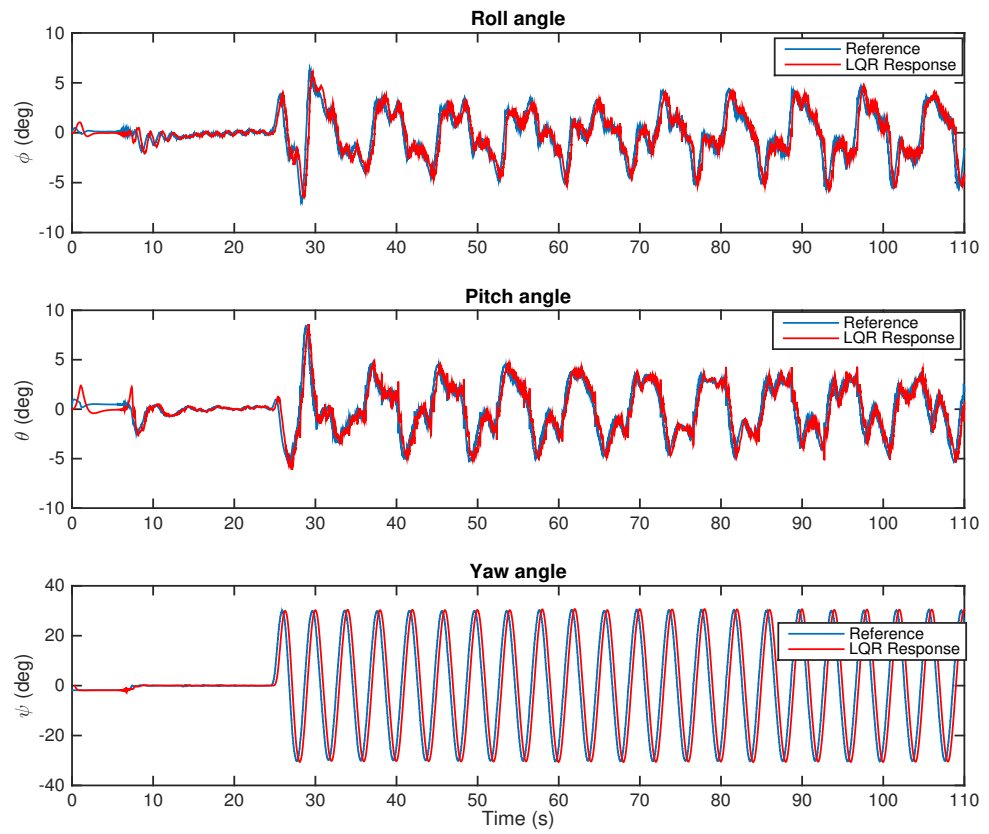


Figure 4.14: Roll, Pitch, and Yaw angle tracking using LQR technique

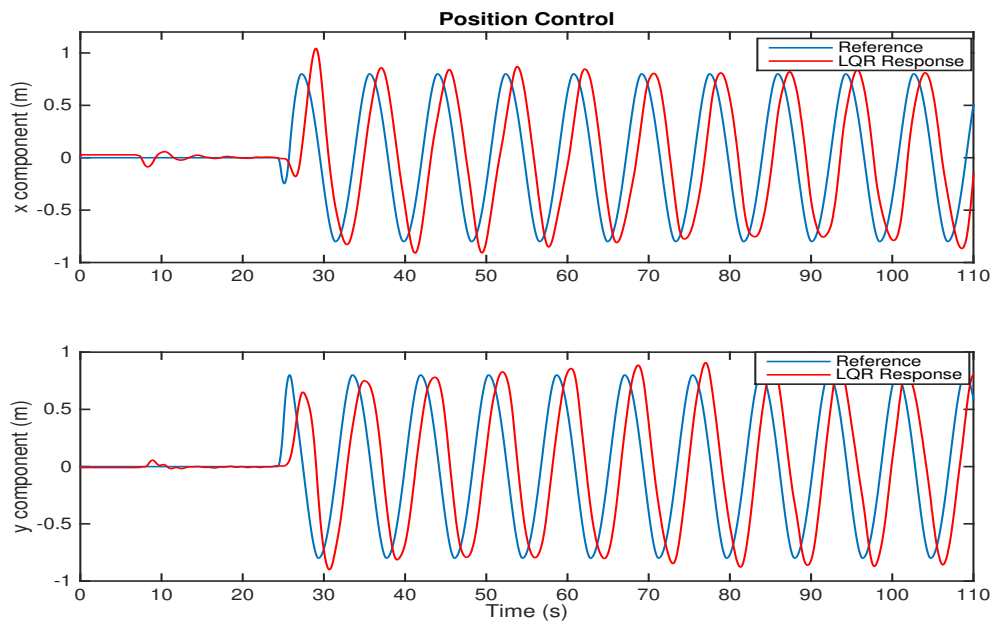


Figure 4.15: Position tracking using LQR technique

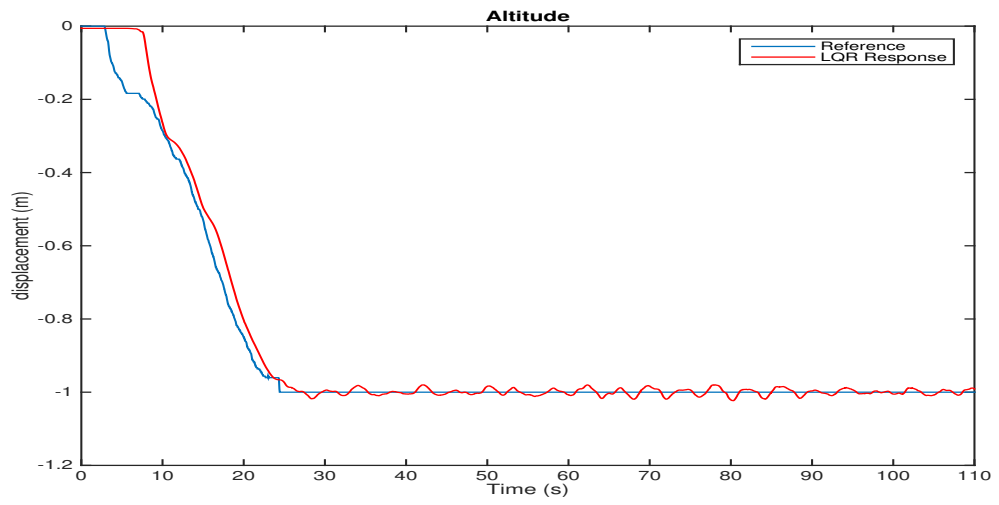


Figure 4.16: Altitude tracking using LQR technique

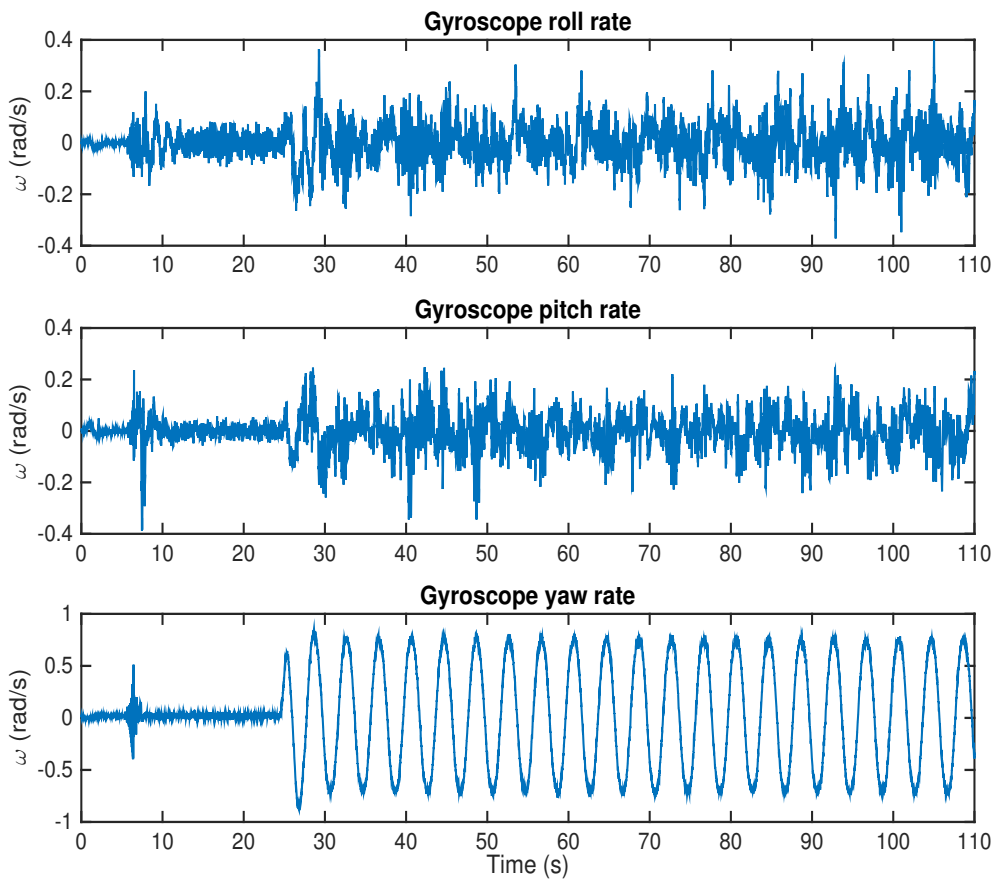


Figure 4.17: Angular Velocities

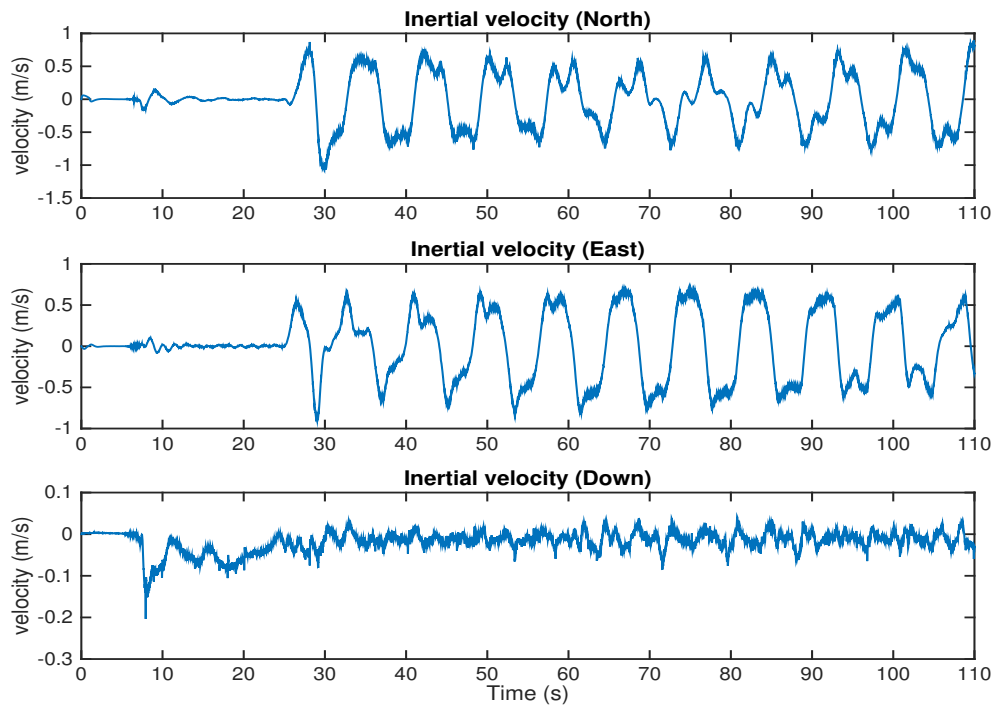


Figure 4.18: Linear Velocities with respect to inertial frame

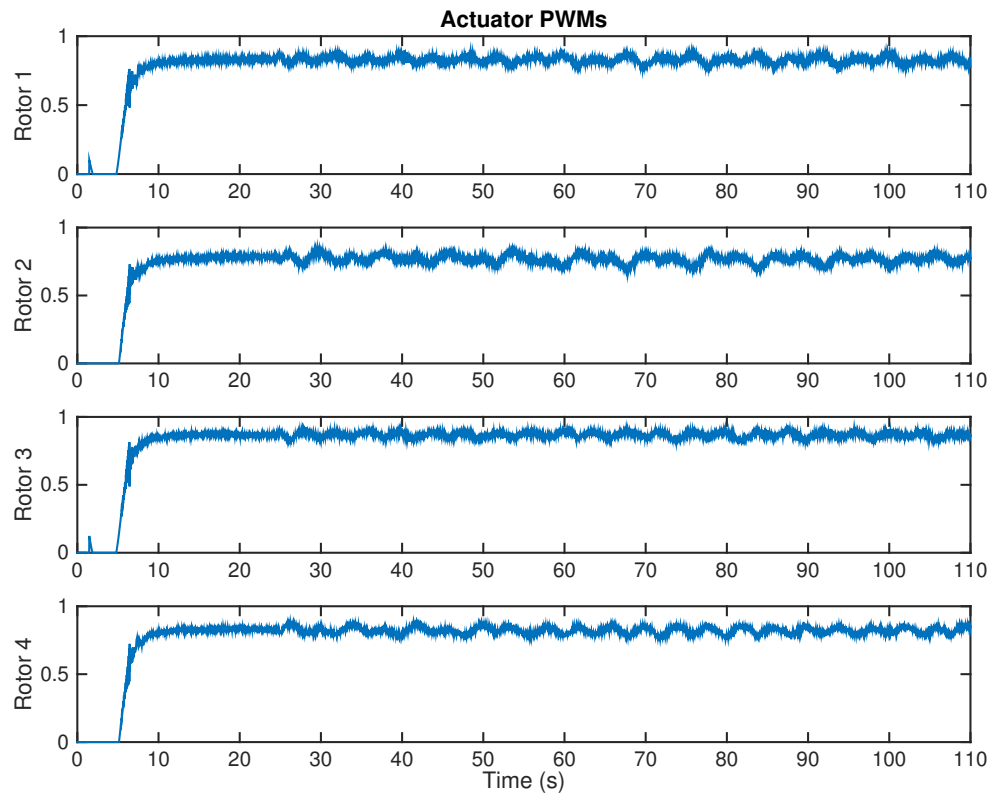


Figure 4.19: Actuator Signals

## 4.6 CONCLUSION

In this chapter, Simulation and real-time flight test results of an LQR controller designed for a linearized system model of a quadrotor is described. The LQR is implemented to control x-y positions, Euler angles, and altitude of a quadrotor during real-time flight. The tracking performance of the designed algorithm is tested for a circular trajectory for both the Simulation and real-time flight test. The results from the real-time flight resembles the Simulation results. During normal circumstances when the x-y position feedback is available, both the inner and outer loop controls are established. In case of a possible position measurements loss, only the inner loop parameters are controlled. As long as the quadrotor is operated close to the trim conditions, the results are satisfactory. However, if the system deviates away from the trim conditions, the LQR controller might fail because it is designed using the linearized system model, which is valid close to the trim values. So, the state reference commands should be restricted to values close to the trim conditions. The measurements required for the Simulation is provided by integrating the nonlinear state equations, whereas, for the real-time flight they are provided by the Extended Kalman Filter explained in Chapter 3.

## 5. OBSTACLE DETECTION AND C MEX S-FUNCTIONS

### 5.1 INTRODUCTION

In this section, an obstacle detection scheme using a laser rangefinder sensor and C MEX S-functions to integrate C programs in a Simulink model are discussed. A 2D laser rangefinder sensor is used to generate a point cloud to show the obstacles present within the range of the sensor. Additionally, a C code in a C MEX S-function is tested with a multi-threading concept implemented. The C library of the laser rangefinder is also transferred to the C MEX S-function file by making necessary changes in order to make it compatible with Simulink. Finally, one of the scan result is saved in a file and plotted to see the distance of the obstacles present in the environment with respect to the sensor origin.

### 5.2 PROBLEM FORMULATION

#### 5.2.1 OBSTACLE DETECTION

A Hokuyo URG-04LX-UG01 laser rangefinder sensor is used to provide distance measurements between the target and the vehicle [37]. It can detect an obstacle anywhere between 20mm to 5600mm and has a scanning range of  $240^0$  with  $0.36^0$  angular resolution. In order to test the sensor and generate appropriate distance measurements, a built-in C library is used [38]. In Figure 5.1, the sensor front is pointed towards the right and sensor left is pointed upwards. The sensor scan starts from the point indicated by 0, whereas it ends at the max index covering  $240^0$ . The index changes from 0 to 681 as the sensor complete one scan. The region which is not shaded blue is not scanned by the laser sensor.

A scan similar to the one shown in Figure 5.1 is obtained by scanning the surroundings of an in-door environment using the laser sensor. In order to generate the map, the C library of the sensor is transferred to a C MEX S-function. To generate distance measurements from the sensor, communication is established, and the distance data for one complete scan of the environment is saved in a file. Using the saved data and after some manipulations, the obstacles present within the range of the laser sensor are plotted based on the sensor axis system.

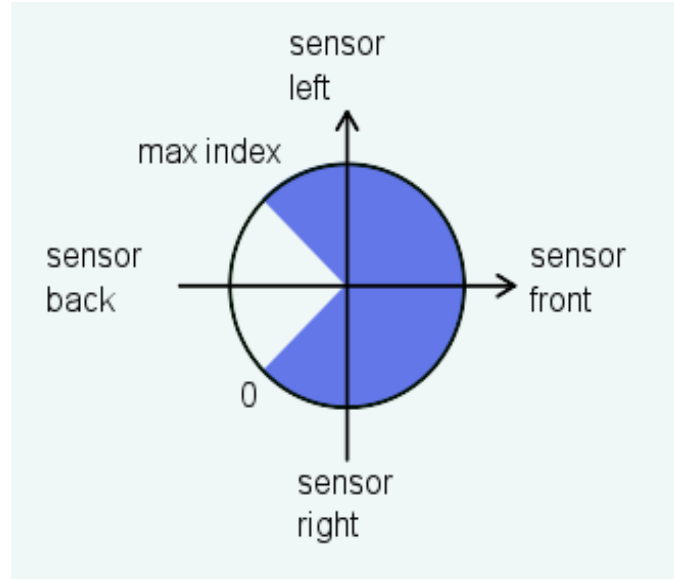


Figure 5.1: Laser Rangefinder Scan [39]

### 5.2.2 C MEX S-FUNCTIONS

The C MEX S-functions are used to integrate C codes with MATLAB/Simulink. The general structure of a C MEX S-function is shown in figure 5.2.

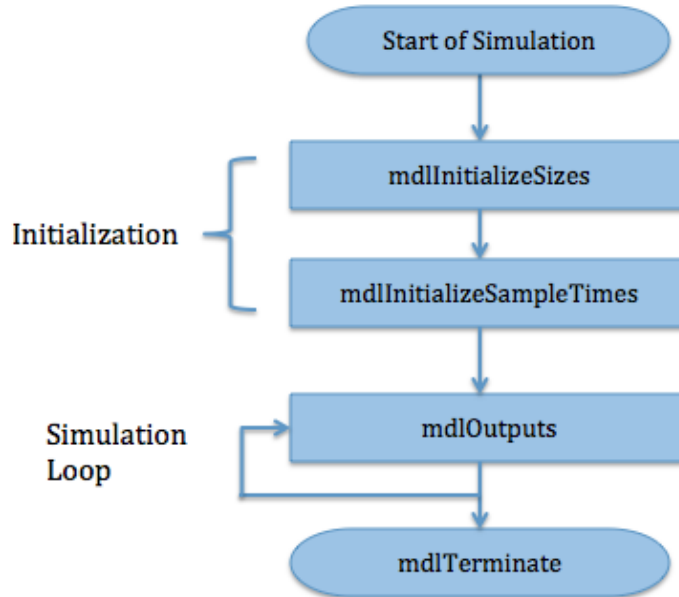


Figure 5.2: Architecture of a C MEX S-function [40]

The initialization blocks specifies the number of inputs and outputs, port size, and the sample time. The output function `mdlOutputs` of the C MEX file is called at each step time to compute the output. Finally, the `mdlTerminate` function is used to perform

tasks at the end of the simulation.

The concept of threading using the pthread library is implemented in the C MEX S-function. The pthread library is included in the C MEX file, and a simple code is written in which a thread object is created in the initialization block which points to a function. The body of this function can be anything based on the requirements, but for testing purposes a variable is counted up in this thread function which is given as an output in the mdlOutputs block. This function can be used to implement complex algorithms (for instance, for the purpose of obstacle avoidance), whose execution can be very time consuming without using threads. Mutexes are also used to lock and unlock the thread in order to guard the data which is required for the output. After the simulation ends, the thread is made to exit/stop by writing the required command in the mdlTerminate block. Figure 5.3 shows a possible experimental model for future with the C MEX S-function containing the required C libraries and the obstacle detection and avoidance algorithm implemented in it, while still using the already developed EKF and LQR techniques for state estimates and control purposes.

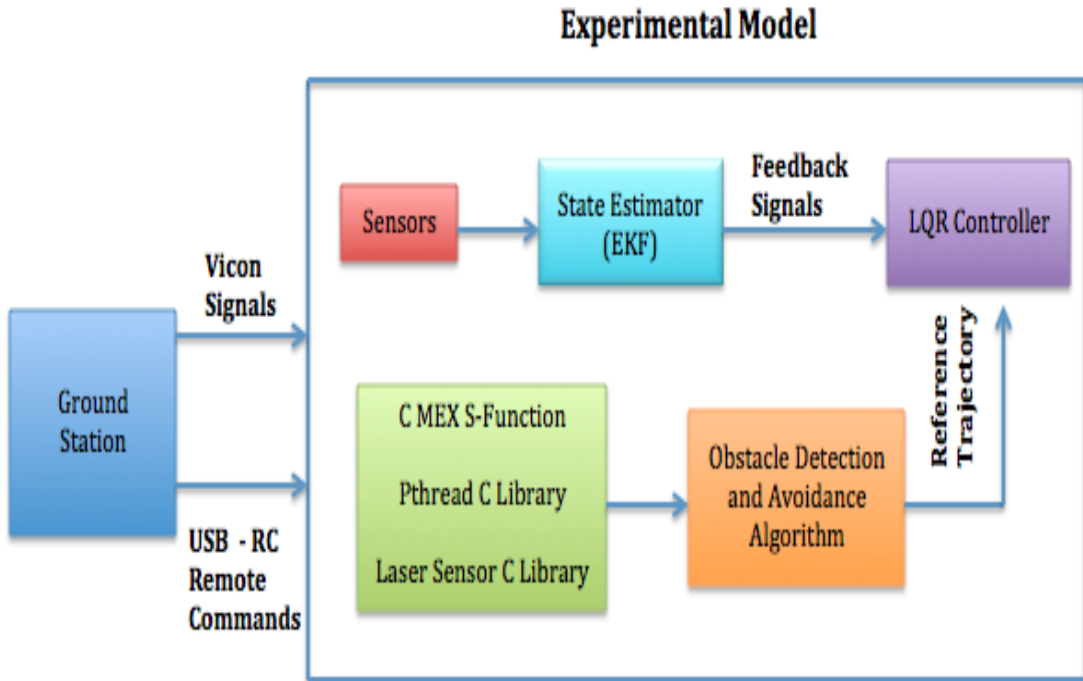


Figure 5.3: Proposed Experimental Model



### 5.3 PRELIMINARY RESULTS

In this section, simulation results are presented using the laser rangefinder sensor distance measurements and the index information to generate the 2D point cloud of the surroundings. This 2D point cloud is then converted to a map showing the size and distance of obstacles. The distance measurements are displayed with respect to the sensor origin and follow the axis system shown in Figure 5.1. Figure 5.4 shows the 2D point cloud in terms of x and y coordinates in which the red dots represent the distance of the obstacle in terms of x and y coordinates for one complete scan of the laser sensor. Additionally, using this 2D point cloud and the Euclidean distance between the data points, a 2D map shown in Figure 5.5 is created to determine the size of different obstacles in the test environment. The obstacles are randomly placed in the work environment, and the 2D map created as a result of laser sensor resembles the test environment, which confirms the accuracy of the sensor and the successful integration of the sensor C library in the Simulink C MEX S-function.

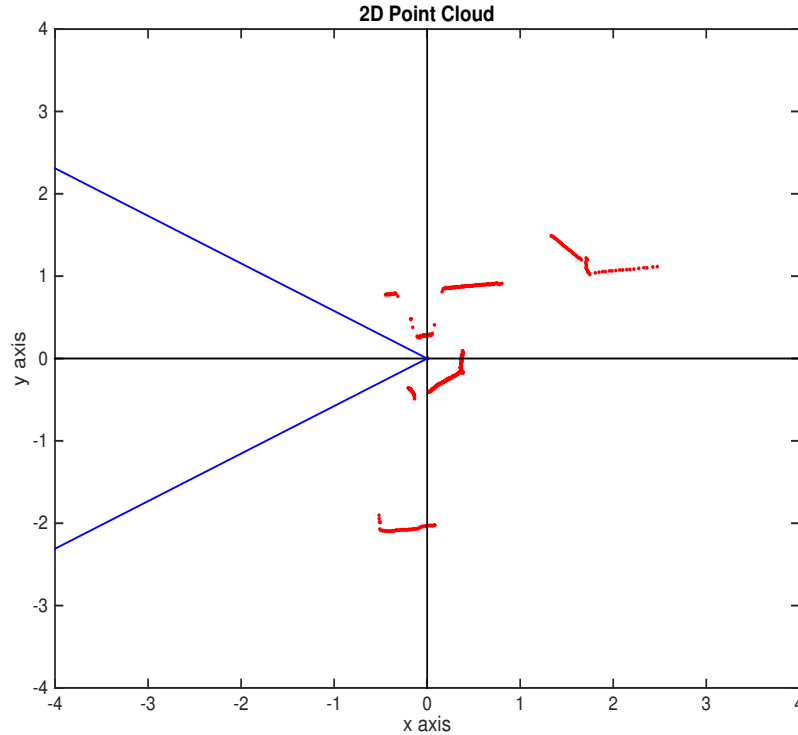


Figure 5.4: 2D Point Cloud of the surroundings

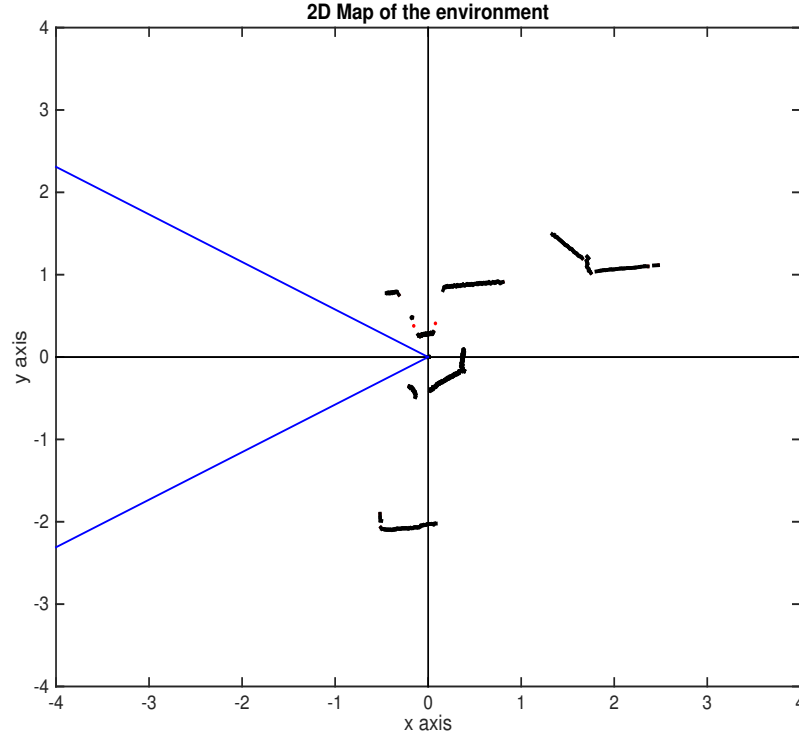


Figure 5.5: 2D Map of the surroundings

#### 5.4 CONCLUSION

In this chapter, C MEX S-functions are used to integrate C libraries and programs with a Simulink model. The laser rangefinder sensor C library is transferred to a C MEX S-function to get the distance measurements and generate a 2D point cloud and map of the environment. The 2D point cloud and map can be used to easily identify obstacles present within the range of the sensor based on the x and y coordinates from the sensor origin. Additionally, pthread library is also integrated with the C MEX S-function, which will later be used to perform computationally expensive tasks related to obstacle detection and avoidance algorithms.

## 6. CONCLUSION AND FUTURE RESEARCH

### 6.1 SUMMARY OF RESULTS

UAV quadrotors have seen increasing growth in the past in military and civil applications, and researchers have been choosing it as a platform to undergo enormous research in the field of robotics, autonomous vehicles, and flight control algorithms. In order to enhance the autonomy and reliability of quadrotor UAVs, advanced control techniques are required to enhance system performance and safety. Inspired by these challenges, this thesis presents the design and real-time implementation of the Extended Kalman Filter to estimate the states of the quadrotor and a Linear Quadratic Regulator (LQR) controller with integral action to satisfy the control objectives.

Two different cases are considered for estimating the states using the Extended Kalman Filter. First, a full-state Extended Kalman Filter is implemented, which estimates all the quadrotor states along with the IMU sensor biases/faults. In this case, Vicon motion camera system is used to provide position and yaw angle measurements. Second, another Extended Kalman Filter is implemented which estimates only the roll and pitch angles, altitude and vertical velocity of the quadrotor. The available measurements to this EKF are only the altitude, yaw angle, and the IMU. In the first case, both inner and outer loop control is established, whereas only the inner loop control is established in the second case. The effectiveness of the designed algorithm for the two EKFs designed are tested on a real-time test environment.

In order to fulfill the control objectives of the quadrotor, an LQR controller with integral action is implemented. Both simulation and real-time flight results are presented in this thesis to verify the performance of the LQR controller. The feedback signals required for the LQR controller are provided by the EKF generated state estimates. Both inner and outer loop controllers are established using the LQR technique. During normal circumstances, (i.e., when position measurements are available and estimates coming from the first EKF are used), both inner and outer loops are controlled in which the roll and pitch references are generated from the x and y position control. In the event that the position measurements are assumed to be lost, the outer loop control is neglected and only the Euler angles and altitude are controlled, meaning only the inner loop variables/states are controlled. In this case, the roll and pitch references are given as direct command, and the feedback signals come from the state estimates from the

second EKF.

Finally, some preliminary results on the integration of C programs with Simulink C MEX S-functions is described. A 2D map of the environment using a laser rangefinder sensor is generated to identify the obstacles present within the range of the sensor. This is accomplished by transferring the C library of the laser sensor to C MEX S-functions. Multi-threading and the integration of pthread library with Simulink using C MEX S-function are also described, which can later be used to perform computationally expensive tasks related to obstacle detection and avoidance algorithms.

## 6.2 FUTURE RESEARCH

During the second EKF implementation, the altitude and yaw reference signals are obtained from the Vicon camera system, but practically speaking on-board sensors like magnetometer and sonar sensors should be used for yaw and altitude measurements. These sensors can be added in the future and integrated with the EKF and controller, so that the algorithm only uses on-board sensors. A possible choice for the yaw measurement can be HMC6343 magnetometer [41], which can be integrated with the available controller using the  $I^2C$  communication protocol. Altitude measurements can be provided by installing a LV-MaxSonar ultrasonic rangefinder sensor [42].

The capabilities of the system can be increased by implementing an obstacle avoidance algorithm for autonomous flight. In order to accomplish this task, first a path generation and tracking algorithm needs to be devised which can take the quadrotor to any point in the 3-D space based on the defined path. Many different path generation techniques have been introduced by researcher over the years [43–46]. A simple guidance technique, such as waypoint guidance by line of sight as described in [47] is under construction for the quadrotor system described in this thesis. In this algorithm, the line of sight guidance algorithm find the angle between vehicle speed vector and the target. Consequently, the vehicle approaches the target by taking the Euclidean distance between the vehicle and the target. This is done by pointing the vehicle velocity vector towards the target.

After the aforementioned path generation algorithm is integrated with the current system described in the thesis, we will be at a stage to develop obstacle detection and path planning techniques to avoid any obstacles between the quadrotor and the

generated path. Several different types of sensors, including laser rangefinders, stereo cameras, monocular cameras, etc., have been used by researchers to obtain the distance measurements. Among these sensors, laser rangefinders are found to be quite popular because they are not computationally expensive to operate, and distance measurements can be easily obtained without having to develop complex image processing algorithms. Similarly, since we are more interested in detecting the obstacles than the identification of the particular target, so cameras are of less interest to us. For indoor testings, where the range of the laser rangefinder is not of much interest, Hokuyo URG-04LX-UG01 laser range finder sensor can be used to provide distance measurements between the target and the vehicle [37]. The algorithm for obstacle avoidance is under development using the distance measurements from the laser rangefinder sensor, which is implemented as a C MEX S-function file in Simulink [48]. This helps to reduce additional computational load on the current Simulink model used for experimental testing. Finally, a path avoidance algorithm will be developed to avoid any obstacles. For instance, [49, 50] use multiple 2-D laser rangefinders to generate 3-D data, [51] utilizes a monocular vision-based feature estimation for terrain mapping, [52, 53] show high level planning from 3-D maps from stereo to detect the obstacles in the path of the vehicle, and many more. The Kalman Filter and Linear Quadratic Regulator control methods developed in this thesis lay a good foundational stone towards developing more advanced navigation and control techniques.

## BIBLIOGRAPHY

- [1] E. Johnson and D. Schrage, "The georgia tech unmanned aerial research vehicle: Gtmax," *AIAA Guidance, Navigation, and Control Conference, Austin*, 2003.
- [2] X. Z. J. Qi and Z. Jiang, "Design and implement of a rotorcraft uav testbed," *IEEE International Conference on Robotics and Biomimetics, Kunming, China*, 2006.
- [3] Massachusetts Institute of Technology Lincoln Laboratory. Unmanned aerial vehicle (uav) systems center overview. [Online]. Available: <https://beaverworks.ll.mit.edu/CMS/bw/uavcenteroverview>
- [4] S. Gupte, P. I. T. Mohandas, and J. M. Conrad, "A survey of quadrotor unmanned aerial vehicles," in *2012 Proceedings of IEEE Southeastcon*, March 2012, pp. 1–6.
- [5] US Dept. of Defense, "The role of autonomy in dod systems," Secretary of Defense, Washington, D.C., Tech. Rep., July 2012.
- [6] R. C. Leishman, J. C. Macdonald, R. W. Beard, and T. W. McLain, "Quadrotors and accelerometers: State estimation with an improved dynamic model," *IEEE Control Systems*, vol. 34, no. 1, pp. 28–41, 2014.
- [7] B. S. M. Henriques, "Estimation and control of a quadrotor attitude," Master's thesis, Instituto Superior Tcnico, Lisboa, Portugal, June 2011.
- [8] B. J. Emran, M. Al-Omari, M. F. Abdel-Hafez, and M. A. Jara-dat, "A cascaded approach for quadrotor's attitude estimation," *Procedia Technology*, vol. 15, pp. 268 – 277, 2014. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S2212017314001959>
- [9] A. Wu, E. Johnson, and A. Proctor, "Vision-aided inertial navigation for flight control," *AIAA Guidance, Navigation, and Control Conference*

- and Exhibit, Guidance, Navigation, and Control*, 2005. [Online]. Available: <http://dx.doi.org/10.2514/6.2005-5998>
- [10] “Flight pid controller design for a uav quadrotor,” *Scientific Research and Essays*, 2010.
  - [11] G. V. Raffo, M. G. Ortega, and F. R. Rubio, “An integral predictive/nonlinear h-infinity control structure for a quadrotor helicopter,” *Automatica*, vol. 46, pp. 29–39, 2009.
  - [12] T. Madani and A. Benallegue, “Backstepping control for a quadrotor helicopter,” in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2006, pp. 3255–3260.
  - [13] S. Khatoon, D. Gupta, and L. K. Das, “Pid amp; lqr control for a quadrotor: Modeling and simulation,” in *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Sept 2014, pp. 796–802.
  - [14] S. Bouabdallah, A. Noth, and R. Siegwart, “Pid vs lq control techniques applied to an indoor micro quadrotor,” in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, Sept 2004, pp. 2451–2456 vol.3.
  - [15] B. Fan, J. Sun, and Y. Yu, “A lqr controller for a quadrotor: Design and experiment,” in *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, Nov 2016, pp. 81–86.
  - [16] I. R. Morar and I. Nascu, “Model simplification of an unmanned aerial vehicle,” in *Proceedings of 2012 IEEE International Conference on Automation, Quality and Testing, Robotics*, May 2012, pp. 591–596.
  - [17] R. W. Beard and T. W. McLain, *Small Unmanned Aircraft: Theory and Practice*, 1st ed. Princeton. NJ: Princeton Univ. Press, 2012.
  - [18] J. Diebel, “Representing attitude: Euler angles, unit quaternions, and rotation vectors,” Stanford University, Tech. Rep., October 2006.

- [19] A. Alaimo, V. Artale, C. Milazzo, and A. Ricciardello, "Comparison between euler and quaternion parametrization in uav dynamics," vol. 1558. AIP Conference Proceedings, 2013, pp. 1228–1231.
- [20] J. B. Kuipers, *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality*. Princeton University Press, Princeton, New Jersey, 1999.
- [21] R. Avram, "Fault diagnosis and fault-tolerant control of quadrotor uavs," Ph.D. dissertation, Wright State University, USA, 2016.
- [22] V. M. S. L. UK. Camera systems precise. fast. powerful. [Online]. Available: <https://www.vicon.com/products/camera-systems>
- [23] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME—Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.
- [24] P. S. Maybeck, *Stochastic Models, Estimation, and Control, Volume 1*. Academic Press, Inc, 1979.
- [25] B. Anderson and J. B. Moore, *Linear Optimal Control*. Englewood Cliffs, NJ: Prentice hall, 1971.
- [26] e. A. Gelb, *Applied Optimal Estimation*. Cambridge, MA: MIT Press, 1974.
- [27] R. Lewis, *Optimal Estimation with an Introduction to Stochastic Control Theory*. John Wiley and Sons, Inc, 1986.
- [28] L. ASCORTI, "An application of the extended kalman filter to the attitude control of a quadrotor," Master's thesis, Politecnico Di Milano, Milano, Italy, 2012-2013.
- [29] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3 – 20, 2002. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0005109801001741>
- [30] A. HALL, *The Analysis and Synthesis of Linear Servomechanisms*. The Technology Press, M.I.T. (Cambridge, MA), 1943.



- [31] N. WIENER, *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. MIT Press (Cambridge, MA), 1949.
- [32] G. NEWTON, L. G. JR., and J. KAISER, *Analytical Design of Linear Feedback Controls*. Wiley (New York), 1957.
- [33] M. ATHANS and P. FALB, *Optimal Control*. McGraw-Hill (New York), 1966.
- [34] R. BROCKETT, *Finite Dimensional Linear Systems*. Wiley (New York), 1970.
- [35] B. ANDERSON and J. MOORE, *GLinear Optimal Control*. Prentice Hall (Englewood Cliffs, NJ), 1971.
- [36] H. KWAKERNAAK and R. SIVAN, *Linear Optimal Control Systems*. Wiley (New York), 1972.
- [37] L. HOKUYO AUTOMATIC CO. Distance data output/urg-04lx-ug01. [Online]. Available: <https://www.hokuyo-aut.jp/search/single.php?serial=166>
- [38] S. Kamimura and K. Kimoto. Urg library document. [Online]. Available: <http://urgnetwork.sourceforge.net/html/>
- [39] K. Kimoto. Urg library document (tutorial page). [Online]. Available: <http://urgnetwork.sourceforge.net/html/library-tutorial-page.html>
- [40] I. The MathWorks. Basic c mex s-function. [Online]. Available: <https://www.mathworks.com/help/simulink/sfg/example-of-a-basic-c-mex-s-function.html>
- [41] spart fun START SOMETHING. Sparkfun hmc6343 breakout. [Online]. Available: <https://www.sparkfun.com/products/12916>
- [42] S. Fun. Ultrasonic range finder - lv-maxsonar-ez1. [Online]. Available: <https://www.sparkfun.com/products/639>
- [43] T. Schouwenaars, "Safe trajectory planning of autonomous vehicles," Ph.D. dissertation, Massachusetts Inst. Technol., Boston, Dept. Aeronautics Astronautics, 2006.

- [44] C. Dever, B. Mettler, E. Feron, J. Popovic, and M. McConley, “Nonlinear trajectory generation for autonomous vehicles via parametrized maneuver classes,” *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 2, pp. 289–382, 2006.
- [45] C. Goerzen, Z. Kong, and B. Mettler, “A survey of motion planning algorithms from the perspective of autonomous uav guidance,” *Journal of Intelligent and Robotic Systems*, vol. 57, no. 1, p. 65, 2009. [Online]. Available: <http://dx.doi.org/10.1007/s10846-009-9383-1>
- [46] C. L. Bottasso, D. Leonello, and B. Savini, “Path planning for autonomous vehicles by trajectory smoothing using motion primitives,” *IEEE Transactions on Control Systems Technology*, vol. 16, no. 6, pp. 1152–1168, Nov 2008.
- [47] M. Lizarraga, R. Curry, and G. H. Elkaim, “Flight test results for an improved line of sight guidance law for uavs,” in *2013 American Control Conference*, June 2013, pp. 818–823.
- [48] I. The MathWorks. Creating c mex s-functions. [Online]. Available: <https://www.mathworks.com/help/simulink/sfg/creating-c-mex-s-functions.html>
- [49] S. Thrun, W. Burgard, and D. Fox, “A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping,” in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 1, 2000, pp. 321–328 vol.1.
- [50] D. Hahnel, W. Burgard, and S. Thrun, “Learning compact 3d models of indoor and outdoor environments with a mobile robot,” in *Proceedings of the Fourth European Workshop on Advanced Mobile Robots (EUROBOT01)*, Sep 2001.
- [51] D. Magree, J. G. Mooney, and E. N. Johnson, “Monocular visual mapping for obstacle avoidance on uavs,” in *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, May 2013, pp. 471–479.
- [52] F. Andert, F. Adolf, L. Goormann, and J. Dittrich, “Mapping and path planning in complex environments: An obstacle avoidance approach for an unmanned helicopter,” in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 745–750.

- [53] L. Heng, L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, “Autonomous obstacle avoidance and maneuvering on a vision-guided mav using on-board processing,” in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 2472–2477.